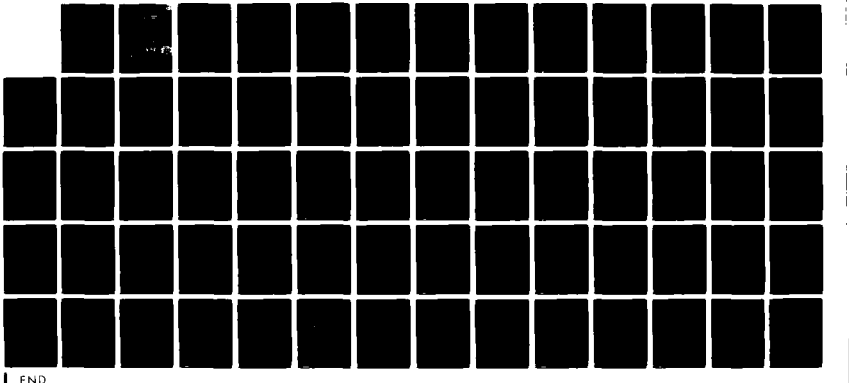
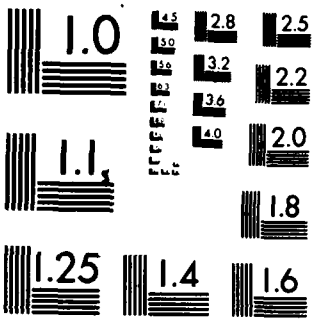
AD-A128 672 STATE-OF-THE-ART ASSESSMENT OF TESTING AND TESTABILITY 1/1
OF CUSTOM LSI/VLSI..(U) AEROSPACE CORP EL SEGUNDO CA
ENGINEERING GROUP A J CARLAN OCT 82
UNCLASSIFIED TR-0083(3902-04)-1-VOL-5 SD-TR-83-20-VOL-5 F/G 9/5 NL



END
DATE FILMED
7-83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A128672

DTIC FILE COPY

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED

83 05 26 10

This final report was submitted by the Aerospace Corporation, El Segundo, CA 90245 under Contract No. F04701-82-C-0083 with the Space Division, Deputy for Logistics and Acquisitions, P.O. Box 92960, Worldway Postal Center, Los Angeles, CA 90009. It was reviewed and approved for The Aerospace Corporation by J. R. Coge, Electronics and Optics Division, Engineering Group. Al Carlan was the project engineer.

This report has been reviewed by the Office of Information and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication. Publication of this report does not constitute Air Force approval of the report's findings or conclusions. It is published only for the exchange and stimulation of ideas.

FOR THE COMMANDER

APPROVED

STEPHEN A. HUNTER, LT COL, USAF
Director, Speciality Engineering
and Test

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SD-TR-8[illegible] 20	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) State-of-the-Art Assessment of Testing and Testability of Custom LSI/VLSI Circuits Vol V:. Design for Testability		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER TR-0083(3902-04)-1
7. AUTHOR(s) M.A. Breuer & Associates and A.J. Carlan, Aerospace Technical Director		8. CONTRACT OR GRANT NUMBER(s) F04701-80-C-0081 AC F04701-81-C-0082 AC F04701-82-C-0083
9. PERFORMING ORGANIZATION NAME AND ADDRESS M.A. Breuer & Associates 16857 Bosque Dr. Encino, CA 91436		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Space Division Air Force Systems Command Los Angeles, Calif. 90245		12. REPORT DATE October 1982 13. NUMBER OF PAGES 67
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) The Aerospace Corporation El Segundo, Calif. 90245		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Scan-in/scan-out Design Techniques	Structured Design Methods
Level Sensitive Scan Design Testability	Microcomputers
TMEAS	Partitioning
SCOAP	Bit Sliced
Bus Oriented Systems	Selective Control Redundancy

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Designing for testability if needed to reduce costs associated with testing and maintaining electronic systems. Two approaches are considered: 1) modification of established circuits and 2) general design of new circuits where testability is a major consideration. Computer programs TMEAS and SCOAP, developed for evaluating testability in established circuits, are discussed. In the design of new circuits only a few techniques are known that yield highly testable circuits without sacrificing other desirable traits, two, IBM's LSSD method and bit slicing, are discussed.

DD FORM 1473 (FACSIMILE)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Design for testability is motivated by the need to reduce the costs associated with testing and maintaining a digital system over its working life. These costs depend on many interrelated factors which are poorly understood and difficulty to quantify. Major testability considerations include test generation difficulty, test sequence length, test application cost, fault coverage and fault resolution. Testability can also be measured indirectly with much less computational effort in terms of two general circuit properties called controllability and observability. Several computer programs have been written recently that compute controllability and observability measures for a given circuit. These programs provide practical tools for comparing the testability of different designs, and can also be used to indicate testing bottlenecks within circuits. The use of such programs is very limited at present.

Two approaches to design for testability have evolved: ad hoc design rules to improve the testability of a given logic circuit, and general design approaches with testability as the primary design objective. The use of test and control points which attempt to improve local observability and controllability, respectively, is one of the most useful of the ad hoc design guidelines. Suitable sites for test points can readily be determined, and include flip-flop set/reset lines, deeply buried components, points of high fan-in or fan-out such as major buses, and logically redundant subcircuits. The principal limitation on this technique is the small number of extra IO pins or connectors available for testing purposes. Testability can also be improved by restructuring a circuit, for example, by opening feedback loops or other strongly connected subcircuits during testing. Additional important design rules include the avoidance of asynchronous timing, and the provision of a mechanism whereby a tester can override or synchronize with the internal clock of the circuit under test.

Because testability involves many tradeoffs, very few general design techniques are known that yield highly testable circuits without sacrificing other important practical considerations. The most promising of these

are are the scan-in/scan-out methods represented by IBM's LSSD (Level Sensitive Scan Design) technique. The basic idea of scan-in/scan-out is to design a circuit so that its memory elements can be linked together to form a shift register SR during testing. This allows the circuit's state (the contents of SR) to be directly controlled and observed by an external tester. Since access to SR is serial, only one or two extra pins are required. Furthermore, most of the circuit is seen as a (large) combinational circuit, for which test pattern generation is relatively easy. LSSD-type circuits have the disadvantage of requiring rather long testing times; they are also impractical for circuits such as RAM chips that contain thousands of memory elements. Another promising design approach of more limited applicability is bit-slicing. Bit-sliced systems consist of an array of identical elements called (bit) slices. The individual slices are relatively easy to test, and tests for an array can be easily derived from those of a slice.

Over the next five years it is likely that increased attention will be paid to design for testability because of the rapid increases in chip complexity resulting from VLSI technology. The use of computer programs that evaluate the testability of unstructured designs is likely to increase. However, structured design like LSSD and bit-slicing lead to systems that are easy to design and test, and may displace unstructured designs in many applications. Scan-in/scan-out methods like LSSD, and related methods like Selective Control, will become more widely used. They meet some of the major constraints imposed by VLSI technology, and allow current test generation methods like the D-algorithm to be used effectively. Not all circuits are suitable for scan-in/scan-out designs, particularly circuits with very large numbers of memory elements. Different approaches which will probably employ self-testing will be required in such cases.

1. INTRODUCTION

This report surveys the techniques that can be used for designing easily testable logic circuits. The scope of the report is limited to methods that require the use of external test equipment. The design of self-testing circuits (built-in test equipment or BITE) and the use of error-detecting codes will be covered in separate reports. This report is also restricted to logic design considerations; electrical, or mechanical and other aspects of physical design which influence testability are not considered.

The motivation for designing easily testable circuits is to reduce the costs of test pattern generation, fault detection both during manufacture and in the field, and fault isolation and repair. These costs depend on many interacting design factors whose relationship are by no means well understood. Thus the concept of design for testability is quite difficult to quantify. Chapter 2 deals with the problem of measuring testability. The factors influencing testability are discussed, two of which, controllability and observability, have proven to be the easiest to measure. Two computer programs TMEAS and SCOAP which were developed for evaluating testability in practical circuits are discussed in this chapter.

The techniques for obtaining easily testable designs may be divided into two broad classes

(1) Methods for improving the testability of an existing design;
(2) General design methods whose primary objective is ease of testing.

Chapter 3 is concerned with the first of these approaches, and presents a catalogue of design modification methods for enhancing testability, including test and control point insertion and circuit restructuring. These techniques are typically used in heuristic fashion, and their impact on the overall testability of a circuit is difficult to estimate. Chapter 4 deals with general design methods that attempt to construct circuits that are a priori easy to test. These methods employ special circuit structures that simplify test pattern application and response

evaluation. A general and very useful design philosophy called scan-in/scan-out is examined. Another structured approach to design for testability based on bit slicing is also discussed. The report concludes with an extensive bibliography.

2. TESTABILITY MEASURES

The classification of a circuit as easily testable implies the existence of objective criteria for judging testability. In practice, testability measures are needed which can be applied to relatively complex circuits, and which do not require an excessive amount of computation. This chapter reviews and evaluates the main testability measures that have been proposed for digital logic circuits. In Sec. 2.1 the testing characteristics of a circuit that can be used as for measuring testability are discussed in general terms. Section 2.2 considers specific quantitative measures based on the concepts of controllability and observability, which form the basis of some practical computer programs for testability evaluation. The chapter concludes with a brief evaluation of these programs.

2.1 General Measures

Figure 2.1 lists five of the main factors that influence circuit testing. The cost c_G of test generation is the cost of computing the input test stimuli and output response sequences required in the given testing environment. If this test data is obtained via a computer program such as LASAR or the D-algorithm [Breuer and Friedman 1976], then c_G may be equated to the cost of developing and running the test generation program. The test sequence length cost c_L may be measured by the (average or maximum) number of test patterns used or, equivalently, by the (average or maximum) time required to apply these tests to the unit under test (UUT). The cost c_A of applying the tests includes the hardware and software overhead, both in external and built-in test equipment, required to store the test data (if necessary), apply input stimuli to the UUT, and verify the resulting responses. The remaining factors listed in Fig. 2.1 are measures of the effectiveness with which the circuit is tested. Equivalently, they can be regarded as measures of the confidence that can be placed in a test applied to the UUT. The fault coverage e_C refers to the percentage of possible faults that are detectable by a given

- Cost of Test Generation c_G
- Test Sequence Length c_L
- Test Application Cost c_A
- Fault Coverage e_C
- Fault Resolution e_R

Fig. 2.1. Factors determining the testability of a digital circuit.

test procedure. Finally, the fault resolution e_R is the ability of the test procedure to locate or isolate faults to specified parts of the system.

In general, it is desirable to minimize the cost factors c_G, c_L and c_A while maximizing the effectiveness factors e_C and e_R. Unfortunately, there are complex and poorly-understood tradeoffs involved which make it extremely difficult to optimize these various factors simultaneously. Furthermore, it is difficulty to separate the inherent testability of a circuit from the methods used to test it. For example, certain combinational circuits used in coding logic have extremely high test generation cost c_G if the standard D-algorithm is used [Goel 1981]. Tests generated by the D-algorithm are typically characterized by low values of c_L, but high values of c_A since the test equipment must store all the test data. On the other hand, provided the number of input lines n is moderate, say $n \leq 32$, any combinational circuit can be tested by exhaustively applying all 2^n possible input patterns. While this tends to maximize the cost c_L, it greatly reduced c_G because the required input patterns can be generated during the testing process itself by a simple hardware or software counter. Figure 2.2 summarizes the effect of these representative test generation methods on the testability requirements of combinational circuits.

In addition, note that for exhaustive testing the costs c_G and c_L are not influenced by the circuit structure, which is not true for test generation using the D-algorithm. Also adding one more input to a circuit would double the value of c_L for exhaustive testing, but would probably have a small impact on c_L when using the D-algorithm.

The foregoing example illustrates the difficulty of quantifying testability or reducing it to a simple universal formula. In many situations, testability is most usefully defined in general quantitative terms, the following definition being a good example [Anon. 1980]:

> Testability [is] a design characteristic which allows the status (operable, inoperable or degraded) of a unit (system, subsystem, module or component) to be confidently determined in a timely fashion.

The same authors define design for testability as follows:

Testability Factor	Testing Method	
	D-algorithm	Exhaustive
Test Generation Cost c_G	very high	very low
Test Sequence Length c_L	fairly low	very high
Test Application Cost c_A	high	fairly low
Fault Coverage e_C	good	very good
Fault Resolution e_R	good	very good

Fig. 2.2. Impact of two testing methods used for combinational circuits on various testability factors.

A design process such that deliberate design effort is expended to assure that a product may be thoroughly tested with minimum effort and cost, and that high confidence may be ascribed to the test results.

In the following section we review some more specific quantitative measures of a design's testability.

In principle, the various testability measures listed in Fig. 2.1 can be computed for arbitrary circuits by actually generating the test stimuli and responses for the circuit, and (exhaustively) analyzing this test data to obtain parameters such as the fault resolution e_R. In practice, the cost of this approach is prohibitive, particularly if a number of different implementations of a given function are to be evaluated. A more practical approach is to determine some circuit properties that (a) can be computed at reasonable cost, and (b) provide a reasonable indication of overall testability, or pin-point parts of a circuit that may be difficult to test. This is the basis of several practical testability measurements developed in the last few years and discussed in the following section.

2.2 Controllability and Observability

Consider the task of generating a test sequence T for a fault F associated with a particular internal line or (signal) node N in a complex circuit C. To be a test for F, the test T must satisfy the following two constraints:

(1) It must allow the tester to control the state of N so that signal values that sensitize C to the presence of F can be applied to N.

(2) It must allow any error signals produced at N to be observed at a primary output line of C. This requires T to establish conditions permitting an error signal to propagate from the fault site N to an observable output.

Thus the controllability and observability of N provide useful indications of the influence of N on the testability of C. The testability measurement methods considered here all attempt to quantify node controllability and observability, and use average or maximum values of these parameters to estimate overall circuit testability.

The use of heuristic controllability and observability measures to guide test generation methods of the D-algorithm type was suggested about ten years ago [Rutman 1972]. Rutman's scheme has recently been extended by Breuer for a proposed functional test generation program called TEST/80 [Breuer 1979]. Three cost values cA, $c\bar{A}$ and dA are associated with each line A that is an output of a component E. cA and $c\bar{A}$ denote the cost of setting A to 1 and 0, respectively, while dA is the cost of driving an error signal D from A to a primary output of the circuit under test. cA is defined by the equation

$$cA = \min\{cfA + csA + cdA, K\}$$

where cfA and cdA depend on the type of the component E, and csA is a "side effects" cost due to fan-out from A. K is a large default value. $c\bar{A}$ and dA are defined similarly. Algorithms have been developed to compute these cost functions for all standard components; these can be used to compute the costs associated with all the lines of a circuit. Although this work has not been implemented, it appears to have strongly influenced the work of Goldstein, which is discussed in detail later in this section [Goldstein 1979]. Theoretical work on computing circuit signal probabilities in combinational circuits is also closely related to the use of controllability and observability measures [Parker and McCluskey 1975, Azema et al. 1977, Dussault 1978]. The formulation of controllability and observability functions in terms of the Boolean difference has also been proposed recently [Susskind 1981].

TMEAS

Stephenson and Grason of Carnegie-Mellon University developed a testability measurement method intended for logic circuits described at the register-transfer level [Stephenson and Grason 1976]. This work was continued by Grason at Bell Laboratories, and resulted in a computer program for testability evaluation called TMEAS [Grason 1978]. The basis of this approach is to associate an input controllability value CY and an output observability value OY with each component and line of the circuit C under consideration. CY and OY are real numbers between 0 and 1. A primary input line has the maximum controllability CY = 1, while a primary output

line has the maximum observability OY = 1. The CY and OY values of all lines are computed by combining rules that view controllability as "flowing," and becoming diminished as it flows, from the primary inputs to the primary outputs of C, while observability flows in the opposite direction. In order to determine the effect of a particular component on controllability/observability flow, two quantities called CTF/OTF (controllability/observability transfer function) must be computed. Somewhat complex formulas were obtained by Stephenson and Grason for exact calculation of CTF and OTF. These transfer functions are basically measures of the uniformity of the input-output relations of the component. In the subsequent TMEAS program more easily computed approximations to these functions were used.

$$CTF = \frac{1}{NOG} \sum_{j=1}^{NOG} \left(1 - \frac{\sum_{i=0}^{NVOG_j - 1} |NIGO_{ij} - NIV/NVOG_j|}{2(NIV - NIV/NVOG_j)} \right)$$

where

NOG = Number of groups (e.g., buses) of output lines

$NVOG_j$ = Number of allowed values on output group j

NIV = Number of allowed input values

$NIGO_{Ij}$ = Number of input values for which output group j has output value i

Fig. 2.3. Formula developed by Stephenson and Grason for calculating a component's controllability transfer function CTF.

Figure 2.3 shows the exact formula for CTF devised by Stephenson and Grason; a similar formula is used to compute OTF. In the case of combinational components, CTF and OTF can be computed directly from the component's truth table. For example, if we apply the CTF formula to an n-input NAND gate we obtain

$$NOG = 1$$

$$NVOG_1 = 2 \qquad (j = 1)$$

$$NIV = 2^n$$

$$NIGO_{01} = 1, \quad NIGO_{11} - 2^n - 1$$

from which it follows that

$$CTF = 1/2^{n-1} \tag{2.1}$$

This very small controllability is indicative of the difficulty of producing a 1 on the output of a NAND gate. It also implies that an (n+1)-input gate is less controllable than an n-input gate, a questionable property of the CTF. THe CTF and OTF of a sequential component is calculated by breaking its feedback loops to convert it into a (pseudo-) combinational circuit. Once the CTF and OTF of a component are known, the controllability and observability of its input/output lines can be calculated as follows:

$$CY_{outputs} = CY_{inputs} \times CTF \tag{2.2}$$

$$OY_{inputs} = OY_{outputs} \times OTF$$

Special circuit models are needed to handle wired logic and bidirectional lines.

TMEAS can provide testability data for board-level circuits quite rapidly; e.g., 3 secs. of CPU time were required on an Amdahl 470 V/7 to analyze a circuit having 70 IC's [Grason 1978]. It has been used to identify good locations for test points (indicated by values of CY or OY near zero), and measure the relative improvement in testability resulting from the insertion of the indicated test points. TMEAS is currently available only for internal use at Bell Laboratories.

SCOAP

More recently, Goldstein of Sandia Laboratories developed another technique for characterizing the controllability and observability of a digital circuit [Goldstein 1979]. This has been implemented in a computer program called SCOAP (Sandia Controllability/Observability Analysis Program) [Goldstein and Thigpen 1980]. Six functions are used to characterize each line or signal node N of the circuit; three functions indicate the difficulty of controlling or observing N from a combinational viewpoint, while the other three employ a sequential viewpoint. A node is referred to as combinational if it is a primary input line, or an output line of a combinational component such as a gate. It is called sequential if it is an output line of a sequential element, like a flip-flop. The function $CC^0(N)$ and $CC^1(N)$ are related to the minimum number of combinational nodes whose values must be assigned in order to produce a 0 or a 1, respectively, at N. CO(N) is related to the number of combinational node assignments required to propagate a logic signal from N to a primary output Z of the circuit, and also to the number of components between N and Z. The three sequential functions $SC^0(N)$, $SC^1(N)$ and SO(N) are defined analogously, with sequential nodes replacing combinational nodes in the preceding definitions.

The six SCOAP controllability/observability functions "flow" through a circuit in much the same way as the CY/CO measures of Stephenson and Grason. The values at any primary input line X are defined by the equations:

$$CC^0(X) = CC^1(X) = 1$$

$$SC^0(X) = SC^1(X) = 0$$

while at a primary output line Z we have

$$CO(Z) = SO(Z) = 0$$

To determine any of the controllability function values of an output line Y of a circuit component or "standard cell," all possible input assignments that give the desired value on Y are examined, and the sum of the corresponding controllability function values of the inputs is computed. The minimum of these sums incremented by a number representing the cell depth is taken as the controllability of Y. Similarly, the observability of an input node W of a cell is taken to be the sum of the following three numbers: the minimum value of the corresponding observability function which appears on some cell output node Y, the minimum sum of the cell input controllability values required to sensitize Y, and the cell depth. Goldstein provides somewhat heuristic rules for computing these functions, which are analogous to the CTF/OTF functions of TMEAS.

As an example, consider an n-input NAND gate G with inputs $X_1, X_2, \ldots, X_n$ and output Y. The combinational and sequential depths of G are assumed to be 1 and 0, respectively. In order to apply 0 to Y, all inputs must be 1, hence

$$CC^0(Y) = \sum_{i=1}^{n} CC^1(X_i) + 1 \qquad (2.3)$$

$$SC^0(Y) = \sum_{i=1}^{n} SC^1(X_i)$$

Y can be set to 1 by setting any of the n inputs to 0, therefore

$$CC^1(Y) = \min_i\{CC^0(X_i)\} + 1 \qquad (2.4)$$

$$SC^1(Y) = \min_i\{SC^0(X_i)\}$$

If the inputs of G are all primary input lines then Eqs. (2.3) and (2.4) become

$$CC^0(Y) = n + 1$$

$$CC^1(Y) = 2$$

The corresponding controllability measure given by Eqs. (2.1) and (2.2) for TMEAS is

$$CY(Y) = 1/2^{n-1}$$

The observability functions for each input line X_i of G are calculated from the following formulas

$$CP(X_i) = CO(Y) + \sum_{\substack{j=1 \\ j \neq i}}^{n} CC^1(X_j) + 1$$

$$SO(X_i) = SO(Y) + \sum_{\substack{j=1 \\ j \neq i}}^{n} SC^1(X_j)$$

which reflect the fact that, in order to sensitize Y to changes in X_i, all input lines X_j where $j \neq i$ must be set to 1.

SCOAP was programmed on a DEC-10 computer, and typical run times of 3.5 min for a 200-cell circuit were obtained [Goldstein 1979]. Figure 2.4 shows an example of a 7-stage feedback shift register analyzed by this program. The results obtained are shown in Figs. 2.5 and 2.6 in the form of circuit "profile" diagrams. These are intended to characterize overall testability of the circuit, and pin-point regions of low controllability or observability for possible improvement. The various controllability or observability function values are plotted on the horizontal axis, while the number of nodes having a particular value are

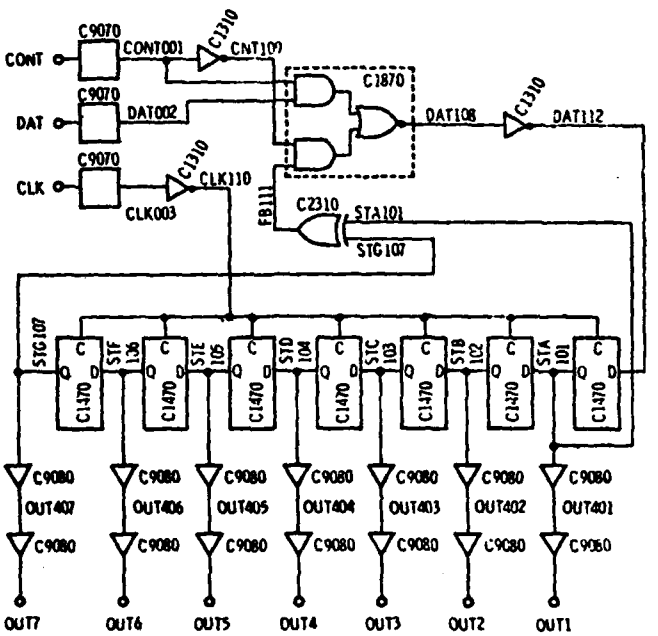
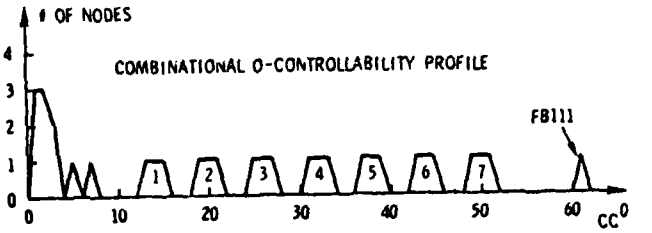
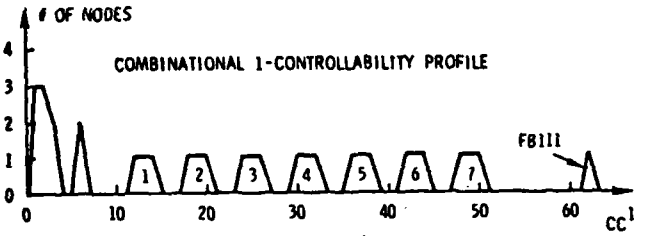


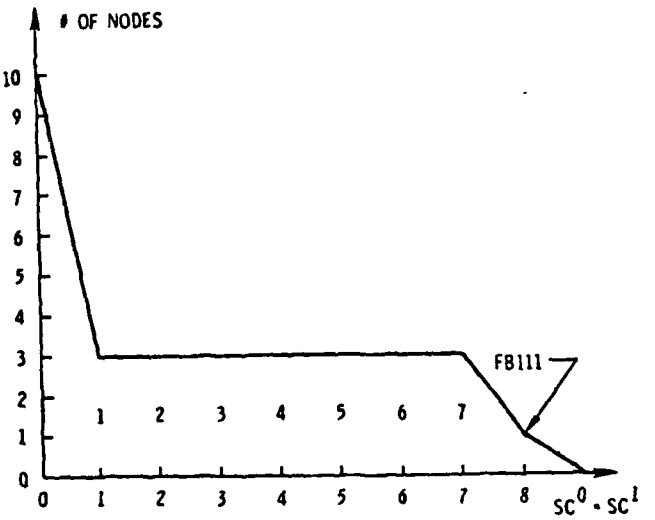
Fig. 2.4. A 7-stage feedback shift register used as a test circuit for SCOAP.



(a)

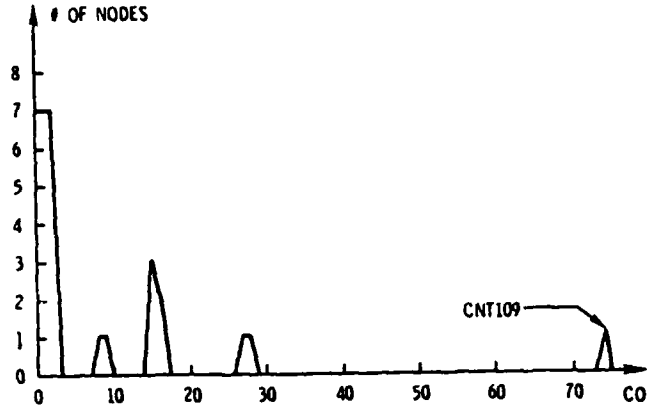


(b)

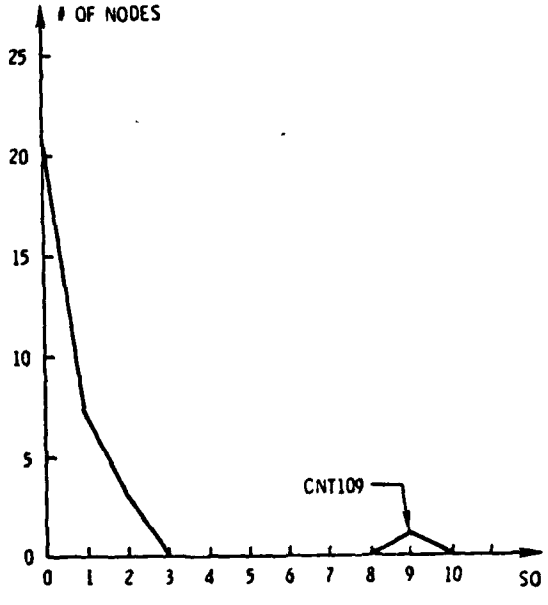


(c)

Fig. 2.5. Controllability profiles for the 7-stage feedback shift register: (a) $CC^0(x)$, (b) $CC^1(x)$, (c) $SC^0 = SC^1(x)$.



(a)



(b)

Fig. 2.6. Observability profiles for the 7-stage feedback shift register: (a) CO(X), (b) SO(X).

plotted on the vertical axis. As expected, the points appearing on the left of the controllability diagrams of Fig. 2.4 represent nodes close to the primary inputs of the circuit. The worst case occurs for the node FB111 which is driven by a feedback path of maximum length; this is also as expected. The observability profiles of Fig. 2.5 indicate that the deeply-buried node CNT109 is significantly less observable than the other nodes, suggesting that it is a good candidate to be a test point.

The overall testability of a circuit can be measured by the maximum values of the various controllability and observability functions. For example, in the circuit of Fig. 2.3

$$CC_{max} = 62$$

$$CO_{max} = 74$$

These values can be recomputed to measure the improvement in testability resulting from various design modifications.

2.3 Summary and Evaluation

The approaches used in TMEAS and SCOAP rely on heuristic but well-defined measures of node controllability of a digital logic circuit. These measures are primarily useful for measuring the relative testability of different designs. Since they provide testability measures for the individual nodes of a circuit, they can be used as a guide to the design modifications needed to enhance testability. For example, an unusually bad controllability figure indicates a possible site for adding a control input line. It may also indicate the location of an undetectable or hard-to-detect fault. Similarly, bad observability figures can also indicate suitable sites for adding output (test) points as well as the location of faults that are difficult or impossible to observe. Besides serving to guide the design modifiction process, programs like TMEAS and SCOAP can be used to guide test generation algorithms. For example, node observability information can be used in a path sensitization process, such as that employed in the D-algorithm [Breuer & Friedman 1976, Roth 1980], to select a path from a fault location to the "most observable" primary output node that can be reached.

The major difference between the two approaches discussed in the preceding section lies in the relative importance assigned to the individual input/output assignment possibilities for a cell. Stephenson and Grason use an averaging method, which appears to be more pessimistic than Goldstein's method where minimum-cost assignments are chosen. For example, in the case of an n-input NAND gate used as an example earlier, Stephenson and Grason obtain an output controllability value $CY(Y) = 1/2^{n-1}$, while the corresponding figures for Goldstein's procedure are $CC^0(Y) = n+1$ and $CC^1(Y) = 2$. The CY figure implies that controllability decreases rapidly as new input lines are added to the NAND gate, whereas the CC figures increase slowly or remain constant as n increases. The CY measures of Goldstein seem to correspond more closely to the intuitive concept of controllability. For example, we would expect $CC^0(Y)$ to increase with the addition of a new input X_{n+1} since X_{n+1} imposes an extra controllability constraint, namely it must be set to 1 in addition to all the previous constraints on $X_1, X_2, \ldots, X_n$ in order to apply 0 to the NAND's output line Y.

From the meager data that has been published, it appears that TMEAS and SCOAP are about equal in performance and usefulness. Each provides similar data on the controllability and observability of the individual nodes of a circuit. Each program can analyze a circuit of moderate complexity in a small fraction of the time required to execute a conventional test generation program. Both TMEAS and SCOAP have been developed into working tools for use by digital system designers; and SCOAP is available to some institutions.

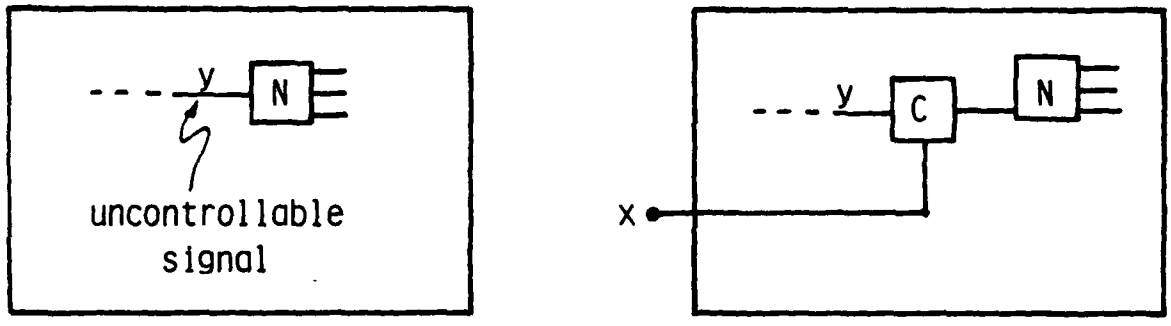
3. DESIGN GUIDELINES

This chapter reviews the techniques available for modifying existing logic designs to improve their testability. Thus testability is viewed here as secondary to the usual design goals of minimizing hardware cost or maximizing operating speed. It also recognizes the fact that testability must often be added to a system after its design is essentially complete. Complete design procedures with testability as their primary goal are examined in Chapter 4.

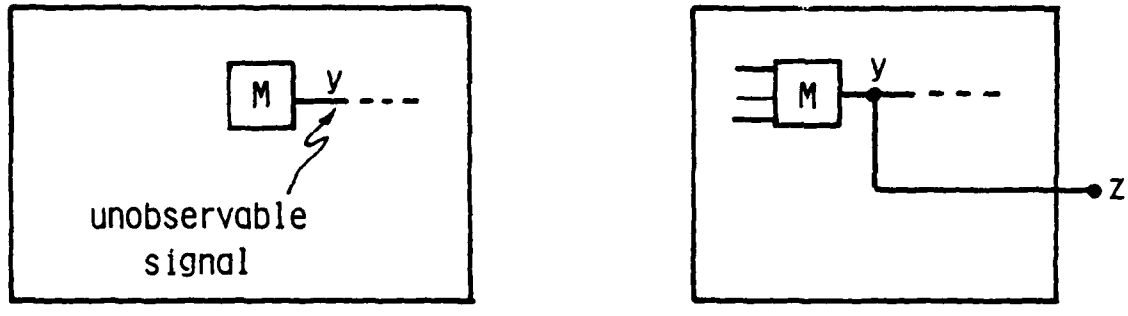
The methods discussed in this chapter are concerned with improving one or more of the testability measures listed in Fig. 2.1. In many cases the goal is approached indirectly by attempting to improve controllability and observability which, as shown in the preceding chapter, are circuit properties that are readily quantifiable. Section 3.1 considers the use of extra input and output lines to increase controllability or observability. In Sec. 3.2 methods of reducing circuit complexity by partitioning a circuit into easily testable subcircuits are discussed. Miscellaneous design issues including timing control are discussed in Sec. 3.3.

3.1 Control and Test Point Insertion

A control point may be defined as an independent primary input line and associated circuits added to a system to increase controllability. Figure 3.1a shows a typical example, where a new input line x and a circuit C are used to control the line y. For example, if C is an EXCLUSIVE-OR gate, x can be used to complement y at will. Similarly, a test point is defined as an independent primary output line and associated circuits added to a system to increase observability. Figure 3.1b shows the simplest case where a new output line z is used by itself to make the line y directly observable. The terms control and test point are often restricted to the added I/O lines. In some cases a single line may act a both a control and a test point, in which case the more general term test point is used.



(a)



(b)

Fig. 3.1. (a) Use of a control point to increase controllability.
(b) Use of a test point to increase observability.

A major constraint of the use of test and control points is the fact that the number of spare external connection points, e.g., IC pins or PCB edge connectors, available for testing purposes is usually very small. This is true at all design levels, from IC chips to complete systems. Thus a key problem here is to select a small number of test or control point sites that yield the maximum improvement in testability. Circuit analysis programs such as TMEAS and SCOAP (see Chap. 2) are very useful tools for this selection process.

Suitable sites for test and control point insertion fall into four major groups.

(1) Flip-flops determining the systems' major control states.
(2) "Long and narrow" subcircuits containing deeply buried components.
(3) "Short and wide" subcircuits having high fan-in or fan-out.
(4) Logically redundant subcircuits containing inherently undetectable faults.

Examples of these cases are considered in the sequel, first as control point sites, and then as test point sites.

Control Point Selection

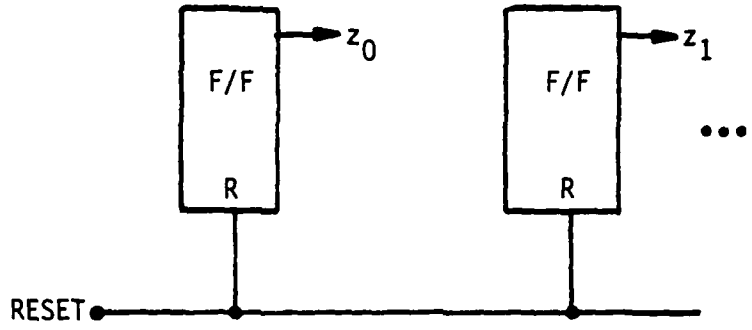
In general, good control point sites are those that are inherently difficult to control unless direct access to them is provided. Corresponding to the first three cases listed above we consider in turn memory elements, deeply buried components, and components with high fan-in as possible locations for control points.

Memory elements in the control unit of a system determine the system's state S; it is extremely important to be able to control S for testing purposes. This requires a mechanism that can initialize S to a known value when testing begins. It has been claimed that almost half of the design problems associated with sequential circuit testing at the PCB level are due to the lack of proper initialization features [Consolla and

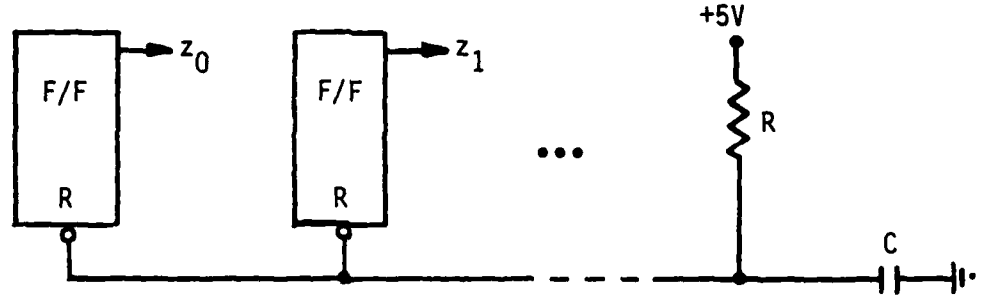
Danner 1980]. Good initialization design should allow the circuit to be quickly set to a known state using relatively little of the main processing circuits which may be faulty.

Figure 3.2 shows several possible approaches. The reset inputs of all the flip-flops defining S can be connected to a common reset line. This allows a single input signal for the tester to make S = 0, and requires only one extra input line. If this extra input line cannot be provided, e.g., because of IC pin limitations, it can be replaced by the RC circuit of Fig. 3.2b which causes a temporary reset signal to be sent to the flip-flops immediately after the circuit power is switched on. If more than one control input can be added, then control over the data inputs of the flip-flops can be obtained by allowing one of several possible initial states to be selected. In the extreme case, parallel access can be provided to the data inputs of all flip-flops allowing any state to be chosen as an initial state; this scheme is usually impractical because of the large number of control lines it requires. If, on the other hand, serial access is provided to the data inputs of the flip-flops, then the system can be initialized to any state using only one or two added inputs. Figure 3.2c illustrates this approach. State information is loaded serially via the D_{in} line, which may be an existing primary input of the circuit. The control line T and the circuits C are used to switch the circuit between the normal mode of operation and a test mode. During the test mode the flip-flops are chained together to form a shift register. While any initial state can be set in the test mode, it requires n clock periods to do so, where n is the number of flip-flops involved. Note that this serial chaining technique is also used in several important general design methods to be discussed in Chap. 4.

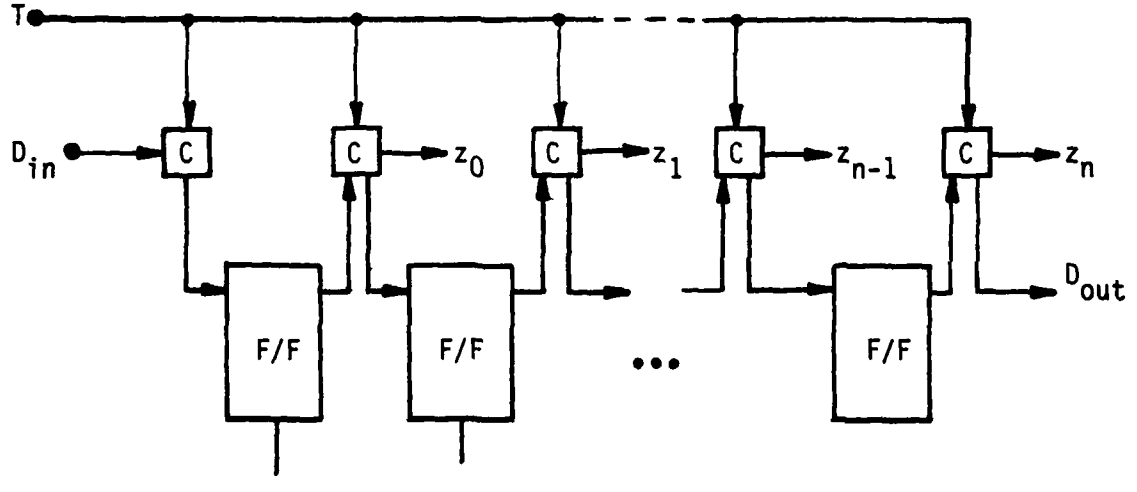
The second type of circuit structure that suffers from poor controllability is a "long narrow" circuit where internal flip-flops are deeply buried in the circuit relative to the controllable primary inputs. Consider, for example, the n-bit counter cirucit appearing in Fig. 3.3a. In order to test this circuit by passing it once through all states, a total of 2^n



(a)



(b)



(c)

Fig.3.2. Various state initialization schemes: (a) Common reset control line; (b) Power-on reset; (c) Serial data reset.

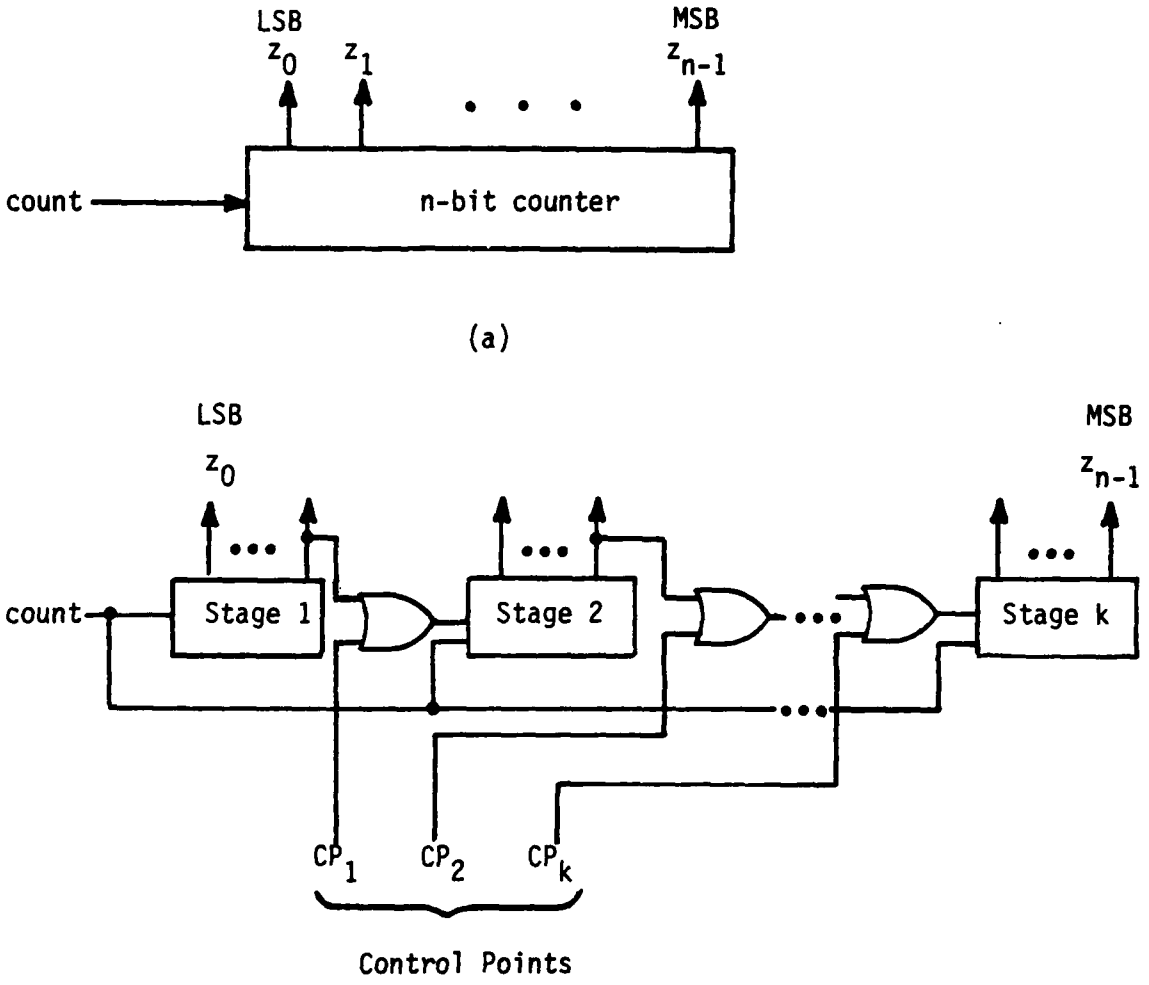


Fig. 3.3 (a) An n-bit counter. (b) Subdivision into n/k-bit counter stages by the addition of k control points.

count pulses must be applied. This may be a relatively large number. Suppose that in the course of test generation, it is required to apply 1 to the most significant bit (MSB) output line z_{n-1}. This may be required to detect a fault associated with z_{n-1}, or to justify a test pattern for some other fault. If the counter is initially sent to the all 0-state, then the test generation program is required to find a sequence of length 2^{n-1} which it must apply to the count input; this is an extremely difficult task for a typical test generation program. In less structured circuits with many levels of flip-flops, test generation can be even more difficult.

Control points can be used to alleviate the foregoing problem in the manner depicted in Fig. 3.3b. Here the counter is subdivided into k identical counter stages, each of n/k bits. A separate control input CP_i to stage i allows a count pulse to be directly injected into that stage by the counter, instead of having to propagate through the preceding i-1 stages. Using this approach the maximum length of a count sequence can be reduced from 2^n to $2^{n/k}$. For example if $n = 32$ and $k = 4$, this is a reduction from approximately 4.29×10^9 to 256, and can represent a significant improvement in testability.

Short wide circuits include those with very large fan-in, which has a negative effect on fault resolution. If many lines fan in to a node N, then the origin of fault signals propagating through N can be very difficult to determine. It is desirable, therefore, to reduce the ambiguity among the signals feeding N. This can be done by attaching control points to N that can be used to block internal signals selectively, or allow signals to be injected by the tester. Figure 3.4 shows the typical use of a control point for this purpose. The AND gate G is inserted into the line feeding node N which has very large fan-in, i.e., it receives signals from a large number of independent sources. G allows a tester to force N to 0 by setting CP to 0. Note that an AND gate is chosen for G rather than an OR gate, since N is driven by a NAND gate G_N whose output under average circumstances will be 1 with a probability of 0.96875. An OR would be more appropriate if G_N were a NOR or AND gate. Some circuit technologies allow the gate G to be dispensed with entirely, by permitting the control line CP to be wire-ORed or wire-ANDed directly to the node N.

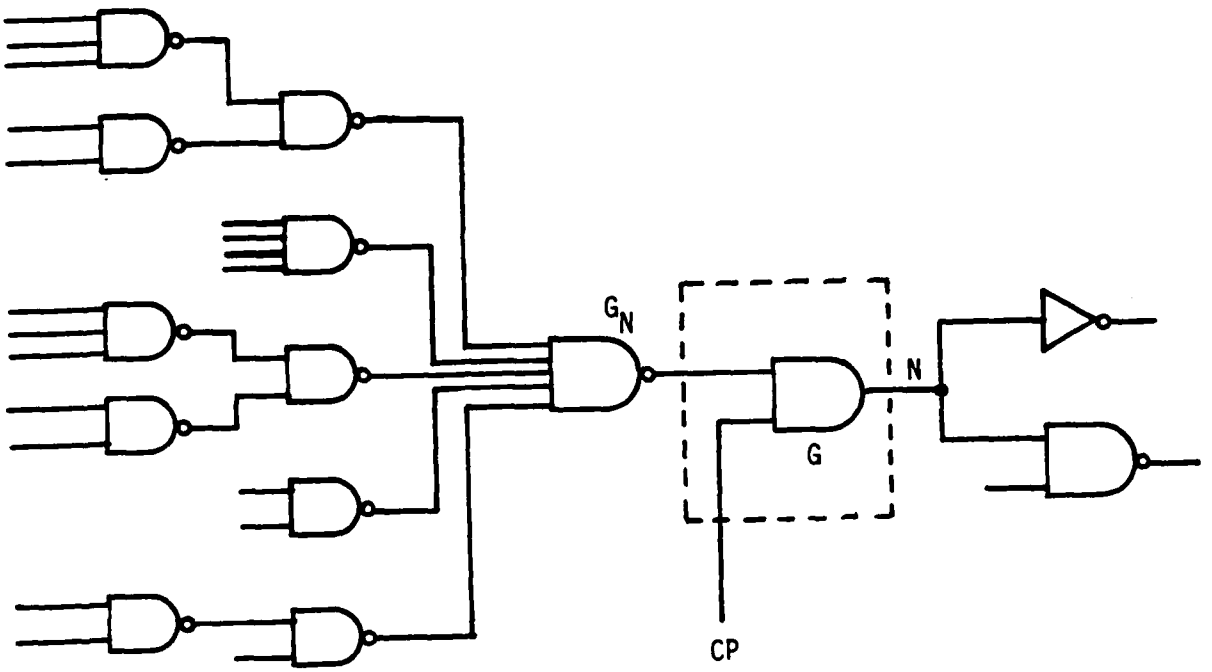


Fig. 3.4. Use of a control point at a circuit node N with high fan-in.

Test Point Selection

Test points are generally inserted at points in a circuit where signal states are difficult to observe. As noted earlier, there are three general design situations where test points are useful: the outputs of memory elements, the outputs of deeply buried components, and components with high fan-out. Test points are usually simple connections from the internal node to be observed to a primary output of the circuit under test. While control input lines can usually be combined and connected to a common input pin, it is much more difficult to combine test points to reduce their pin requirements, since such combination tends to corrupt the information that is being observed [Fox 1977].

The outputs of the memory elements that determine a system's state S are probably the most important points of a system to be made observable during testing. The simplest way to observe S is to connect the memory output signals comparing S to a set of test points. If this is not possible because of pin limitations, then the shift register interconnection scheme of Fig. 3.2c can be used to allow the state of the flip-flops to be shifted out serially via a single test point D_{out} during testing.

The outputs of components that are deeply buried in the sense of having long paths between them and the system's primary outputs, are also good test point candidates. Similarly, lines with very high fanout, like the node N in Fig. 3.5, are also suitable test point sites. Since the signal at N affects many succeeding parts of the circuit, fault resolution can be improved by allowing N to be directly observed during testing. Finally, test points are very useful for making undetectable faults detectable. Such faults typically can exist in redundant subcircuits used for fault masking. By connecting the redundant portion of a circuit to one or more test points, all previously undetectable faults can be made detectable.

Bus-oriented Systems

Many modern systems contain one or more microprocessors and have a global structure in which buses, including data, address and control buses, form the main communication links of the system. These buses are typically

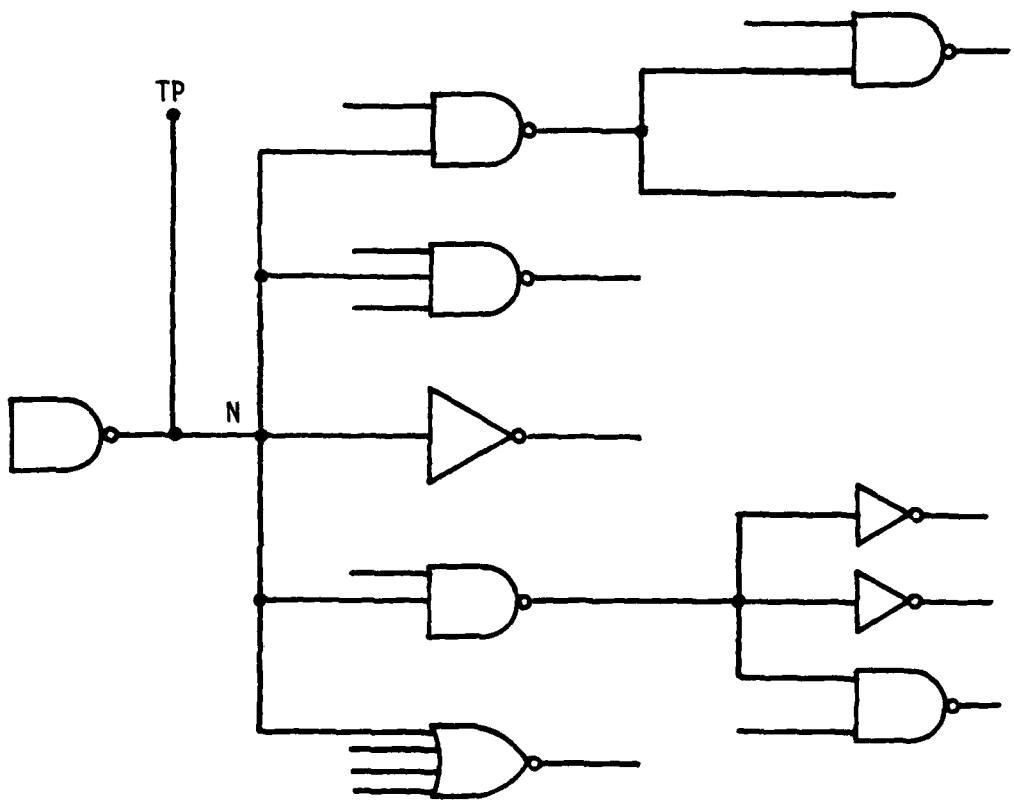


Fig. 3.5. Use of a test point at a circuit node N with high fan-out.

the points of highest fan-in and fan-out in the system. They are therefore extremely desireable control and test point sites. Access to a system bus for control or test purposes can be achieved by channelling the bus to input and output ports of the system, respectively, which are readily used by external test equipment. Special "diagnostic ports" may be provided for this purpose. Alternatively, a regular I/O port may be temporarily connected to a diagnostic port during testing.

This testable design approach is taken in several commercial microprocessor-based devices, for example, the Fairchild/Mostek 3870 one-chip microcomputer [Fairchild 1980]. A single test pin TEST is provided which, when activated, reconfigures the system so that IO port 5 becomes an input to the system's main internal data bus, while IO port 4 becomes an output (test point) for this data bus. In this mode of operation, which is depicted in Fig. 3.6, the internal program ROM is disabled, and the system accepts instructions and operands applied externally via IO port 5. Thus these IO ports constitute a set of control and test points which allow relatively complete access to the microcomputer at the machine instruction level. An extremely important aspect of this type of controllability is that it requires the use of very few dedicated test pins or extra logic. Existing pins (the pins of IO ports 4 and 5 in the present example) provide most of the necessary control and test points.

Another example of the use of existing buses as test points is seen in testing a typical microprocessor such as the Intel 8080 [Chiang and McCorkill 1976]. The CPU's program counter PC can be viewed as the main controlling memory element of the system; its state is therefore very important from a testing viewpoint. The state of PC can be observed during program execution by simply monitoring the system address bus, which is directly connected to PC during instruction fetch machine cycles. An external tester can control PC by jamming an instruction such as NOP (no operation) onto the data bus during an opcode fetch cycle, thereby forcing PC to increment its state by one. By repeating this operation, any desired state can be entered into PC in at most 2^{16} steps.

3.2 Circuit Restructuring

The insertion of test and control points into a circuit can be done in a way that leaves the underlying circuit structure essentially unchanged.

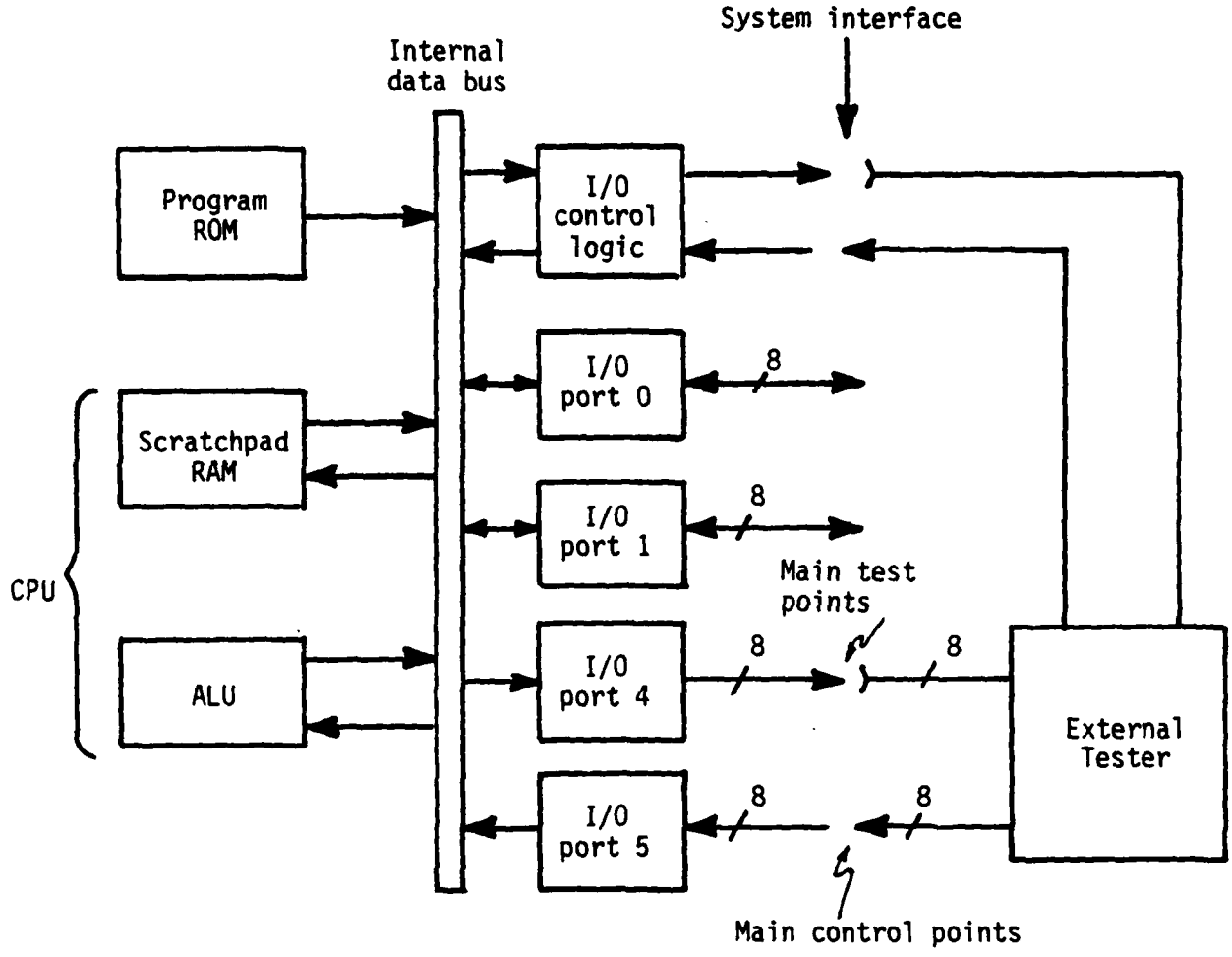


Fig. 3.6. Testing the 3870 one-chip microcomputer.

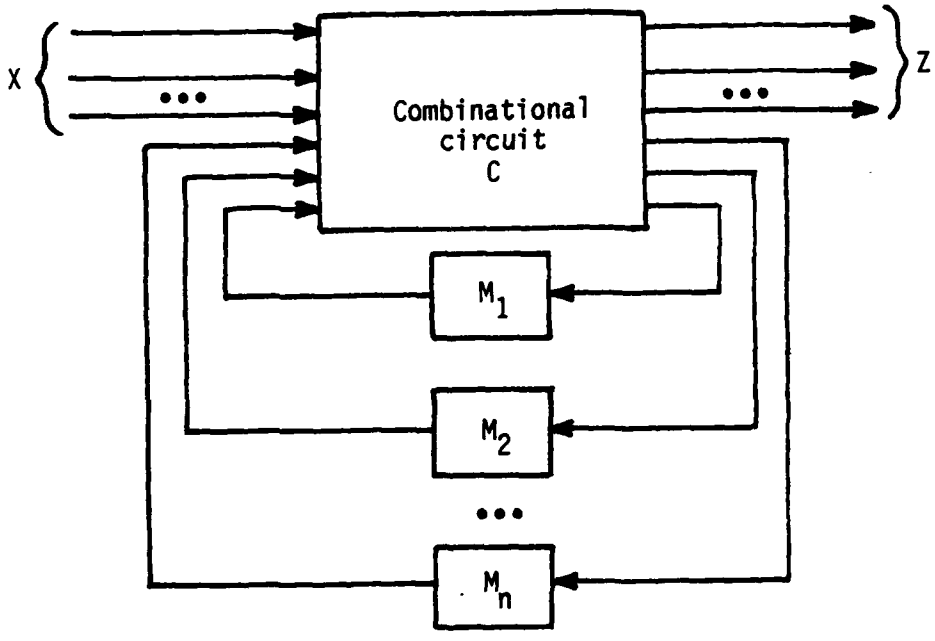
In this section we consider ways in which a given circuit can be made more testable by allowing key aspects of its inherent structure to be altered during testing. Test and control points can play a useful role in this restructuring process.

Feedback Modification

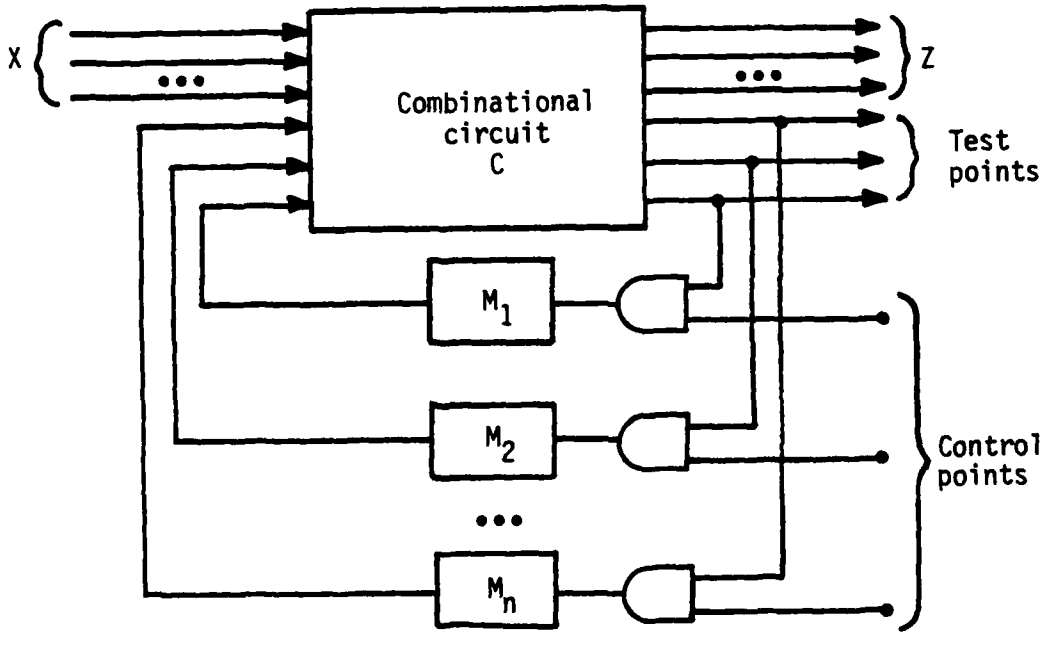
Combinational circuits which contain no feedback are among the easiest to test. Most logic circuits contain feedback of the general form illustrated in Fig. 3.7a. Feedback complicates testing because unanticipated signals can propagate back to the primary input side of the circuit via the feedback loops, thereby interfering with the application of test patterns. A relatively sophisticated test generation procedure is needed to unravel the complexity resulting from the presence of feedback. Moreover, test sequence length tends to grow rapidly with the amount of feedback present in a circuit.

The testing of sequential circuits can thus often be simplified by eliminating feedback, wholly or in part, during testing. This can be done by breaking the feedback loops, either by means of removable jumper wires, or else by means of control points as depicted in Fig. 3.7b. In the latter case, an external control signal is used to insert a logical break in the feedback lines during the test mode of operation. If the feedback lines are also connected to test points as in Fig. 3.6b, then the circuit reduces to a feedforward combinational circuit that is relatively easy to test, and also requires much less testing time than the original sequential circuit. It is also worth noting that many test generation methods such as the D-algorithm are most easily extended to sequential circuits by converting a sequential circuit into an equivalent (pseudo-) combinational circuit.

In systems such as microcomputers (Fig. 3.6) the communication buses form the main feedback paths at the system level. Thus it is generally useful to be able to isolate devices from the buses during testing. This is most easily accomplished by the use of tri-state devices which can be electrically and logically isolated by driving their bus connections to the high-impedance state. It is also sometimes useful to introduce feed-



(a)



(b)

Fig. 3.7. (a) General structure of a sequential circuit. (b) Modification to break feedback loops.

back into such systems to facilitate testing by linking normally disjoint buses. For example, IO ports may be tested by linking their external feedback lines so that they effectively form a closed path. This path can then be tested by writing (output) test signals to one port, and reading (input) response signals from the other port, a method called loop-back [Hayes and McCluskey 1980].

Partitioning

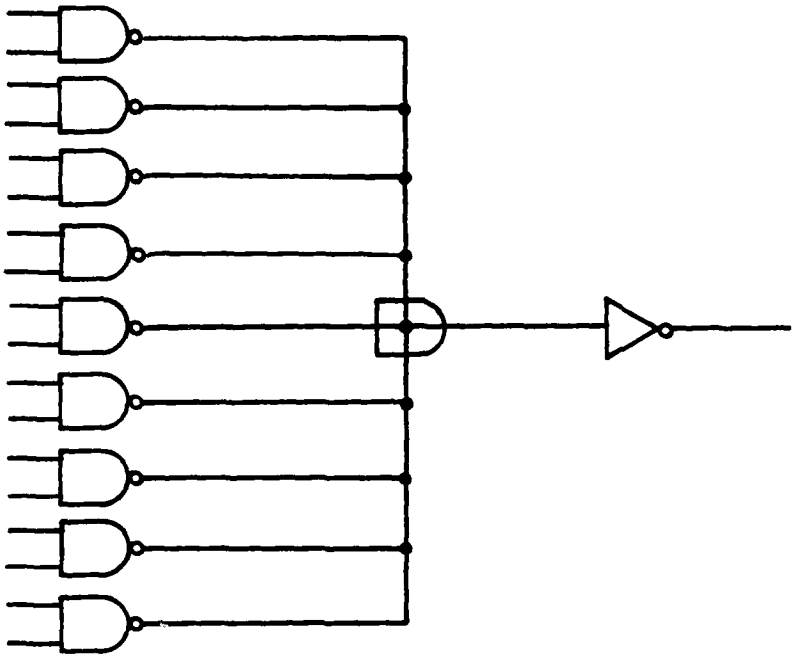
In general, the testability of a complex circuit can be improved by partitioning it into small simply-connected modules. Figure 3.3 shows how a long counter can be made more testable by subdividing it into k short counters, each of which can be controlled independently. This results in a substantial reduction in test sequence length, and also improves fault resolution. Figure 3.8 shows another example of circuit redesign to improve fault resolution. In the circuit of Fig. 3.8a a wired-AND connects the outputs of a large number of NAND gates. This introduces a high degree of ambiguity during fault isolation, since some faults affecting the output circuits of the NAND gates cannot be distinguished. The ambiguity can be reduced by a factor of k or so by arranging the NANDs into k groups as shown in Fig. 3.8b. Each group is connected to the inputs of the added NAND gate G. Since faults affecting the individual inputs of G are distinguishable, we can effectively isolate faults in the original NAND gates to one of the k groups.

A number of general techniques for reorganizing a system design to improve its testability have been proposed. Two representative examples are discussed here.

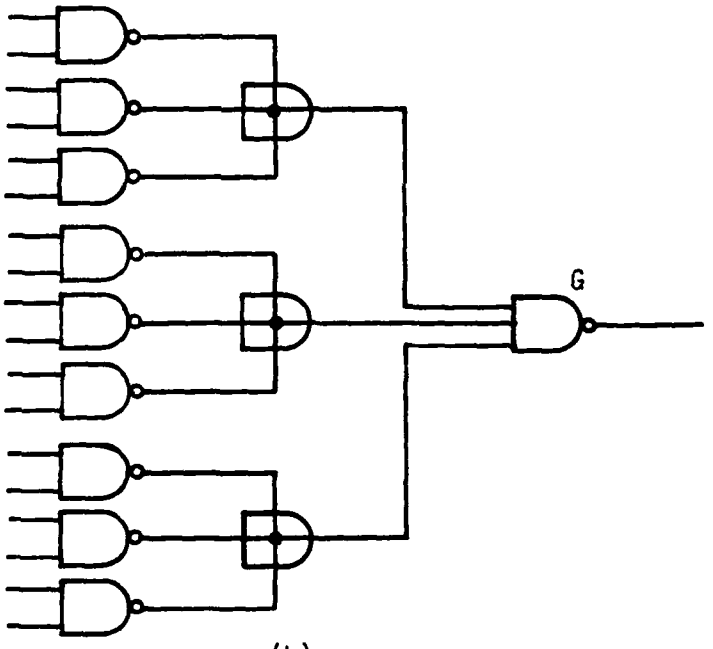
(1) The COMET technique developed at Bell Laboratories [Chang and Heimbigner 1974].

(2) The Selective Control technique developed at IBM [Hsu et al. 1978].

Both are characterized by the fact that the original design is partitioned into relatively small easily-testable subcircuits, and special circuitry is



(a)



(b)

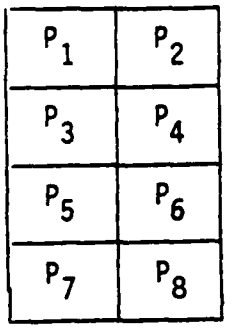
Fig. 3.8. (a) A wired-AND circuit with low fault resolution
(b) Partitioning to increase fault resolution.

inserted at the partition interfaces to provide direct access to the subcircuits. Related recent work includes a partitioning technique for combinational circuits that uses multiplexers as the partitioning devices [Bozorgui-Nesbat and McCluskey 1980].

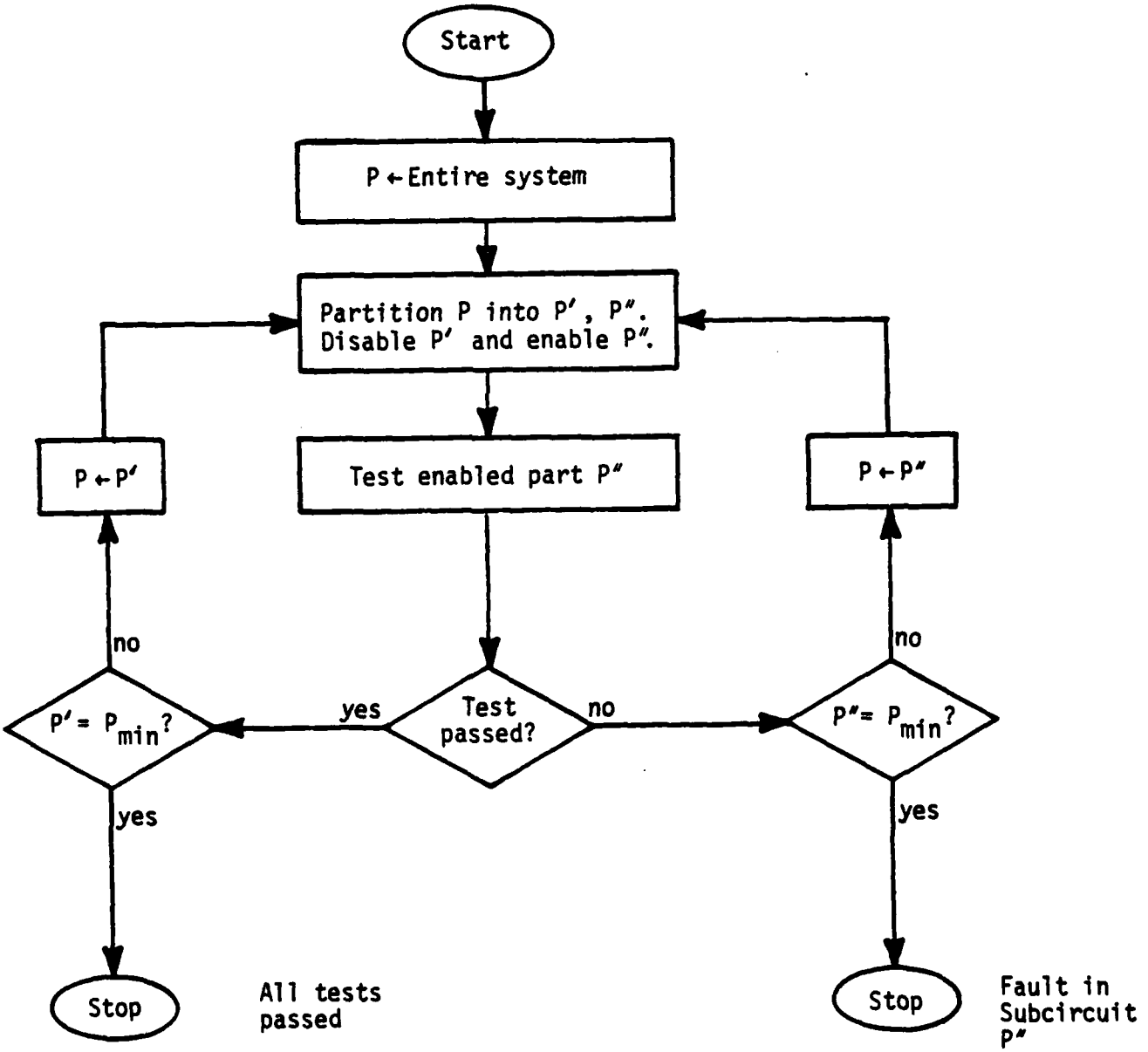
COMET (Controllability, Observability, and Maintenance Technique) aims at enhancing the fault resolution or diagnosability of a system [Chang and Heimbigner 1974]. The basic idea is to partition the system into subcircuits corresponding to minimum-cost replaceable components, e.g., PC boards. Faults are located using the binary search approach depicted in Fig. 3.9. First, half the system, say $P' = \{P_1, P_2, P_3, P_4\}$, is disabled so that it does not affect the testing process. At the same time the rest of the system $P'' = \{P_5, P_6, P_7, P_8\}$ is enabled and tested. Special circuits are added at the itnerfaces of the P_i's to allow P' to be disabled, and to make the inputs and outputs of P'' controllable and observable, respectively. If P'' fails the test, then a fault has been isolated to P''. If P'' passes the test, then the fault, if any is present, must lie in P''. The (potentially) faulty half of the system is then further subdivided into two parts P' and P'', one of which is disabled, while the other is enabled and tested. This process is continued until the partitions corresponding to the smallest replaceable component P_{min} are reached.

The partitioning method used in COMET is based on graph theory. The system under consideration is represented by a directed graph, whose edges indicate the controllability and observability relations among the functional units that constitute the minimal partitions P_{min}. The system graph G is analyzed to identify the maximal strongly connected (MSC) subgraphs which correspond to worst-case sets of nodes that can interfere with one another's controllability and observability. Control and test points are then added to the system in a somewhat ad hoc fashion to break up the MSC's subgraphs. Then the P_i's can be ordered in a way that allows the diagnosis algorithm of Fig. 3.9 to be applied. The added logic for subcircuit disabling, and introducing control and test points is quite simple, typically at most one gate per line as in Fig. 3.1.

COMET was evaluated at Bell Laboratories by a simulation study of a small processor containing a few thousand gates. The results reported indicate that a high degree of fault resolution is achievable, provided a good set of diagnostic tests is available. The amount of extra logic needed to implement COMET is unclear, but is estimated to be less than



(a)



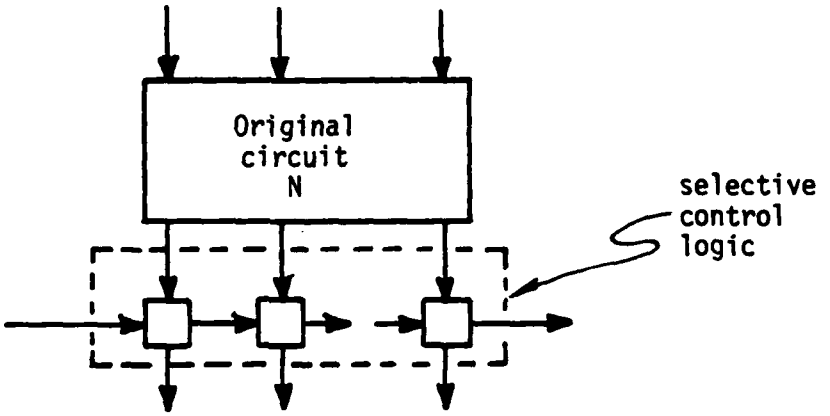
(b)

Fig. 3.9. COMET: (a) Circuit partitions. (b) Diagnostic algorithm.

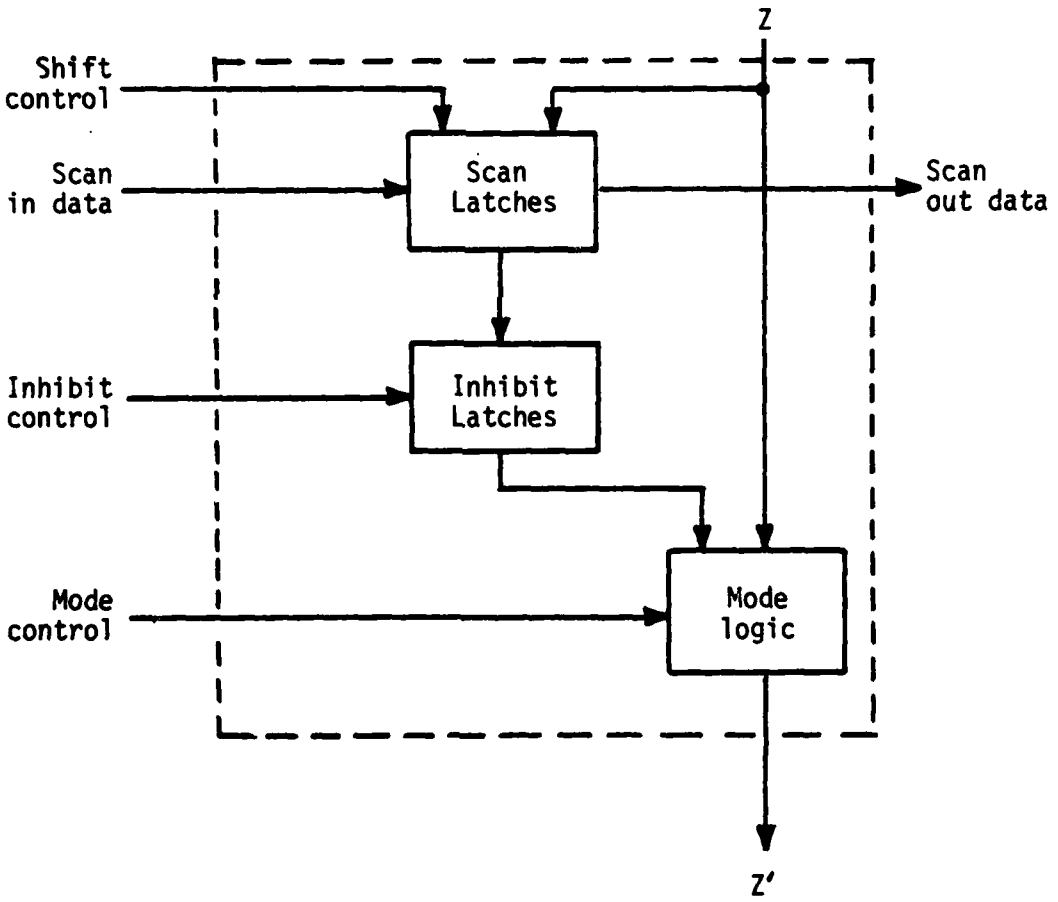
10 percent. COMET does not appear to have been further developed or applied at Bell Laborities beyond this feasibility study.

Selective Control is a technique proposed at IBM to enhance fault diagnosability at the PC board level [Hsu et al. 1978]. Its goal is to use ICs to control the interaction between the ICs of the original circuit and thereby enhance its testability. Specially designed Selective Control (SC) circuits are inserted in the output lines of the original output lines of the original circuit as shown in Fig. 3.10a. They allow these output lines to be selectively inhibited or enabled. The SC circuits contain "scan latches" that can trap output signals from the original circuit for monitoring by the tester. These latches can also be used by the tester to inject test signals into the system that are independent of the original primary input signals. Figure 3.10b shows the internal structure of an SC circuit. The scan latches are interconnected as a shift register, so that they can be accessed serially, thereby minimizing IO pin requirements. Data is transferred from the scan latches to the inhibit latches, thus freeing the scan latches for other purposes. The mode logic allows the new primary outputs Z′ to be driven either from the original primary outputs Z or from the inhibit latches. Z′ can also be driven to the high-impedance state, thus effectively disabling N. A diagnostic procedure similar to that of Fig. 3.9b can be used with selective control.

A major advantage of selective control is that it standardizes the logic circuits that must be added to the original design. The serial access method used for the scan latches allows SC circuits to be chained together in a simple modular fashion. The circuit overhead for selective control is about 20 gates per output line. Also six new I/O connections to the overall system are required. This design technique is closely related to the various scan-in/scan-out design approaches examined in the next chapter. Selective Control can be applied to an LSI chip, where the chip is first partitioned into subcircuits, and the subcircuits interconnected where desired using SC circuits.



(a)



(b)

Fig. 3.10. (a) Circuit with selective control logic. (b) Structure of selective control circuit.

3.4 Miscellaneous Design Rules

We now briefly consider some other important aspects of design for testability.

Redundancy

A subcircuit C is called redundant if it can be removed from the original circuit without affecting the logical behavior of the latter. Redundancy can be present in a circuit for two reasons: it may be an unintentional result of careless design, or it may be inserted deliberately for various reasons. Figure 3.11a shows an example of a redundant circuit used to eliminate a hazard (glitch). The gate G_3 is logically redundant. However, it is included in the circuit to prevent spurious signals from appearing on Z when control shifts between G_1 and G_2; such signals could accidentially reset the flip-flop. Figure 3.11b shows another example of redundancy which is found in fault-tolerant designs. This circuit uses triple modular redundancy (TMR) to allow it to operate properly despite the presence of faults in any of the triplicated units U.

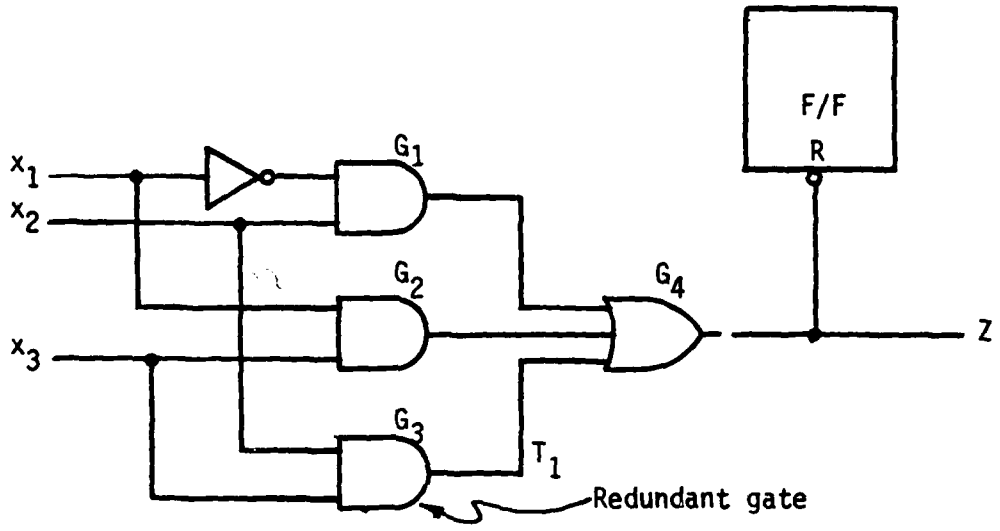
Redundancy, whatever its source, causes two problem from the viewpoint of testability.

(1) It results in undetectable faults, which can result in fruitless and costly attempts by a test generation program to find a test where none exists.

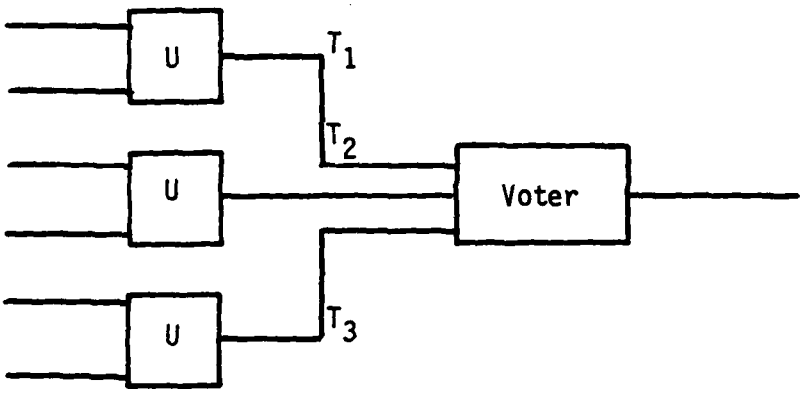
(2) Faults in the redundant part of the circuit may mask detection of otherwise detectable faults elsewhere in the circuit.

In the circuit of Fig. 3.11a, for instance, the fault "output of G_3 stuck-at-0" is undetectable.

In general, care should be taken to avoid unintentional redundancy. If redundancy must be included for one of the reasons cited above, then some provision should be made to allow the redundancy to be removed temporarily during testing. For example, by making the nodes T_1, T_2 and T_3 in Fig. 3.11b into test points, the redundancy is effectively removed. In the case of the TMR circuit of Fig. 3.11b, the interface between the triplicated units and the voter provides an ideal site for selective control logic, which also effectively eliminates the redundancy.



(a)



(b)

Fig. 3.11. Examples of redundant circuits.

Timing Considerations

Asynchronous timing control should be avoided in designing easily testable circuits. Most automatic test generation systems cannot handle the variable delays associated with asynchronous behavior, and awkward "work-arounds" are often needed to handle asynchronism. The test generation problem is also complicated by the need to account for races and hazards, which can be eliminated a priori by the use of synchronous timing. Thus synchronous timing control should be used wherever possible.

Testability can be enhanced by providing an easy way for allowing an external tester to synchronize with, or else bypass entirely, the clock source of the unit under test. For testing the static (timing independent) characteristics of the UUT, it is usually necessary to disable the UUT's internal clock and replace it by clock signals from the tester. Figure 3.12 shows the rather simple circuit needed for this. For dynamic testing it should also be possible to run the UUT at normal operating speed under control of its own clock. To allow an external signal SYNC to synchronize with the UUT, an extra output from the UUT's clock should be provided for the tester as shown in Fig. 3.12.

3.4 Summary and Evaluation

Various general design rules for enhancing testability were encountered in the preceding sections; they are summarized in Fig. 3.13. Because of the relative ease with which controllability and observability can be measured, the insertion of test and control points represent one of the most general and most useful techniques for improving a circuit's testing characteristics. Test and control point selection can be done on an ad hoc basis, or programs like TMEAS and SCOAP can be used to evaluate systematically the potential sites for test and control points. Particularly important sites and key memory elements, deeply buried elements, and elements with large fan-in or fan-out. In microprocessor-based systems, the main buses constitute the most important test and control point sites.

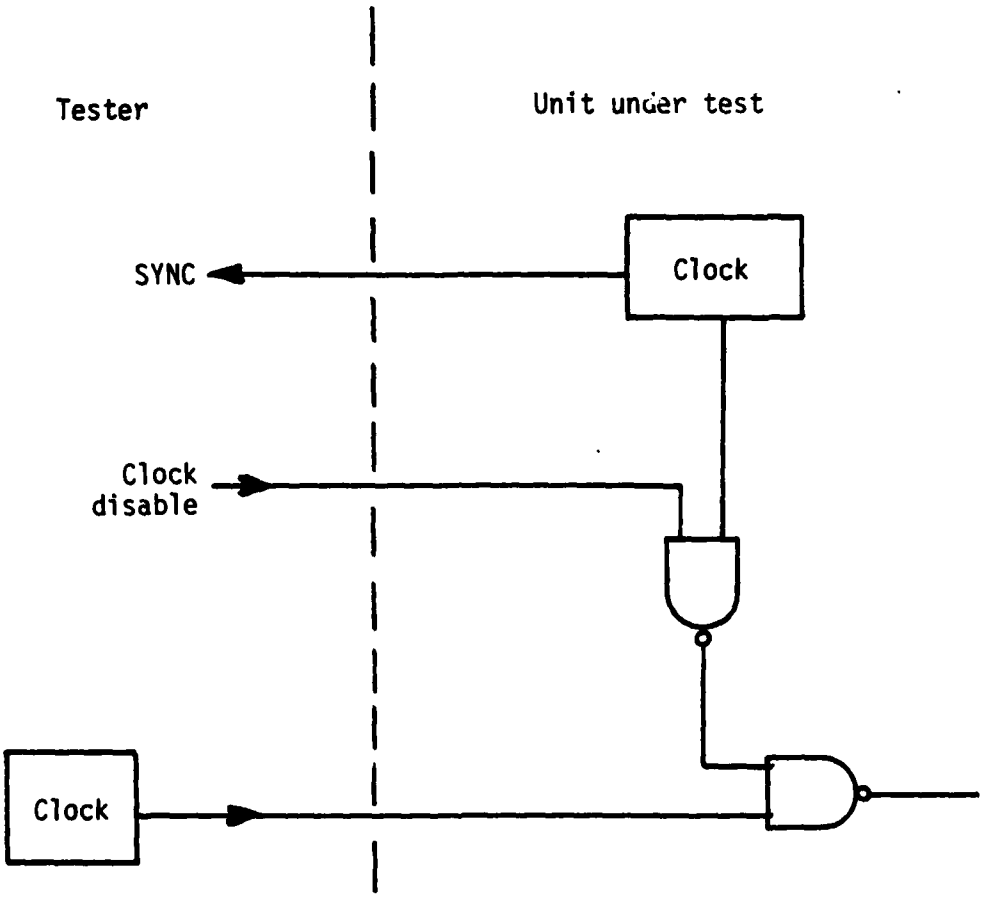


Fig. 3.12. Clock control circuits for testing.

- Design easily initializable circuits
- Avoid redundancy
- Avoid asynchronous timing
- Provide access to UUT's clocks
- Provide means of modifying feedback loops
- Provide access to major buses
- Use control points and test points
- Use wired logic cautiously
- Use partitioning and selective control

Fig. 3.13. Summary of major design rules for testability.

Another general approach is to break the original circuit into simpler subcircuits that can be tested independently. An example of this is the breaking of global feedback loop during testing. Several systematic approaches to circuit partitioning have been proposed, including COMET and Selective Control. COMET is mainly concerned with finding good break points in the circuit. Selective Control, on the other hand, provides a systematic and efficient means of controlling the interfaces between partitions. Its main advantages are the use of uniform interface circuits, and serial access to the interface signals which minimizes the need for additional IO pins.

4. STRUCTURED DESIGN METHODS

In this chapter we examine some complete design methods that aim at producing a system whose testability is known a priori. Testability, therefore, is the primary design objective here, whereas in Chap. 3 testability was viewed as something to be taken care of after the fact.

4.1 Introduction

Because testing involves many conflicting trade-offs, there is no one design methodology that can satisfy all design goals simultaneously. In particular, ease of testing often has an adverse effect on other circuit characteristics. An example of this can be seen in the design method for combinational circuits proposed by Reddy [Reddy 1972]. Reddy's design, which appears in Fig. 4.1, has the very desirable property that all stuck-at-0/1 faults can be detected by at most 3n+4 test patterns, where n is the number of primary imputs. Furthermore, these test patterns are easily derived independently of the function being realized by the circuit. Only one extra I/O line x_0 is required. However, Reddy's circuit is quite impractical for most applications because of the very large number of gates and logic levels it requires.

To be of practical value, a design methodology for ease of testing must satisfy the following criteria.

(1) The overhead in extra logic circuits used for testability should be small, typically no more than 10 to 20 percent of the basic design.

(2) The number of extra IO pins used must be small.

(3) The circuit should achieve normal performance levels when not being tested.

(4) Testing should be carried out using conventional automatic test equipment with little or no special modification.

Very few general design approaches are known that meet all the above requirements. One such method, which we term scan-in/scan-out is discussed

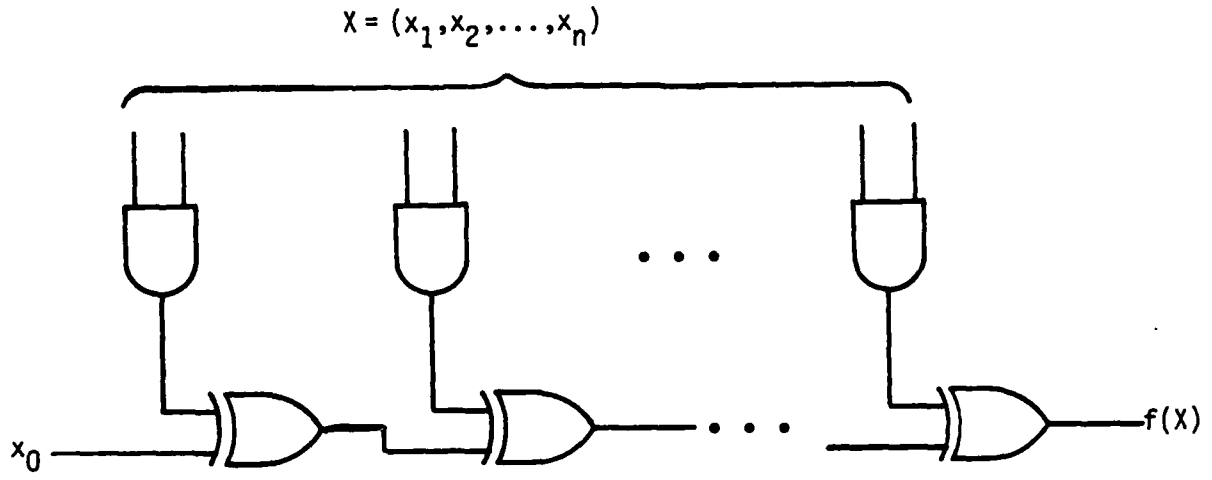


Fig. 4.1. Reddy's easily testable circuit.

in Sec. 4.2. A different appraoch of more restricted applicability based on bit-slicing is discussed in Sec. 4.3.

4.2 Scan-in/Scan-out Methods

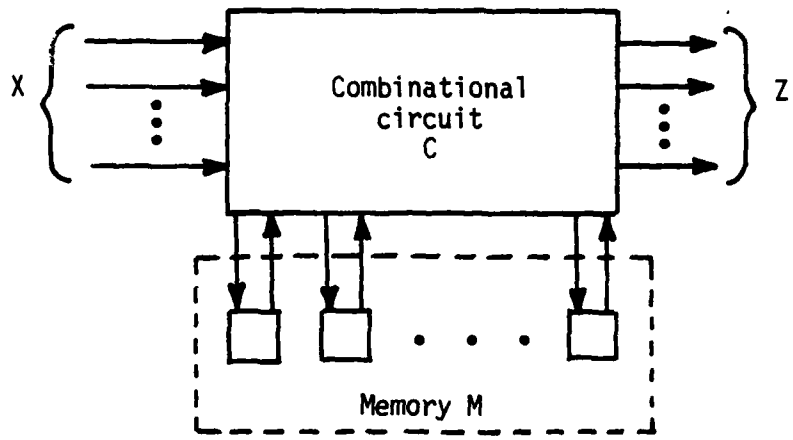
The basic idea underlying this technique is to design a circuit so that all its memory elements (flip-flops) can be linked together to form a shift register during testing, as illustrated in Fig. 4.2. During normal operation the circuit conforms to the usual Huffman model of a synchronous sequential machine. During testing, however, it is effectively partitioned into a (large) combinational circuit C and a shift register SR. The IO lines for C include primary IO lines of the original circuit and connections to SR. Since most of the system's logic resides in C, the major testing problem is to apply a complete set of test patterns to C and verify the resulting responses.

The procedure outlined in Fig. 4.3 is used to apply a test pattern to C and check the result. A special control line P is needed to switch the system between the normal and the test modes of operation; in the latter case the memory elements are configured as the shift register SR. The operation scan-in involves serially shifting into SR a bit stream X_{SR} representing the (input) state of the system. Thus when the system is clocked (Step 4 in Fig. 4.3), it "sees" the state X_{SR} and the primary input pattern X_N. It responds by generating a primary output pattern Z_N, and a next state pattern Z_{SR} which is placed in the system's memory. In the scan-out operation (Step 6), Z_{SR} is shifted serially out of SR for checking.

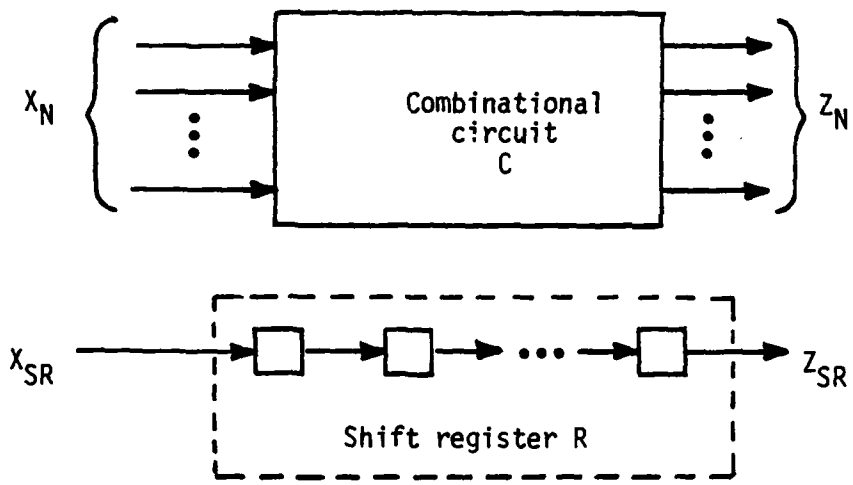
Scan-in/scan-out designs have several very desirable properties.

(1) The test generation problem is reduced to the relatively easy one of testing the combinational circuit C. Efficient methods are known that can generate complete test sets for combinational circuits containing thousands of gates.

(2) The overhead in extra logic is quite small, typically no more than 10 percent. The main requirement for the extra gates is to allow the memory to be configured as a shift register.



(a)



(b)

Fig. 4.2 (a) Circuit during normal operation. (b) Circuit during scan-in or scan-out operations.

1. Set the test mode (P = 1)

2. Shift X_{SR} into SR (scan-in)

3. Set the normal mode (P = 0)

4. Apply X_N and clock the system
   Record Z_N

5. Set the test mode (P = 1)

6. Shift Z_{SR} from SR (scan-out)

7. Verify (Z_N,Z_{SR})

Fig. 4.3. Application of a test pattern using the scan-in/ scan-out method.

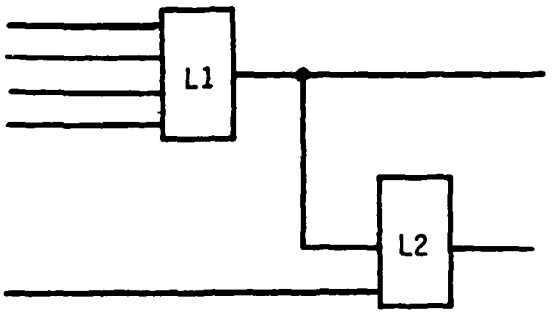
(3) The extra IO pin requirements are minimal. In some implementations of scan-in/scan-out, only a single extra pin is used, namely, the pin P that switches the system between its normal and its test modes.

(4) Apart from the special design of the memory elements, few restrictions are imposed on the circuit design, permitting the usual design objectives pertaining to hardware cost and speed to be met without difficulty.

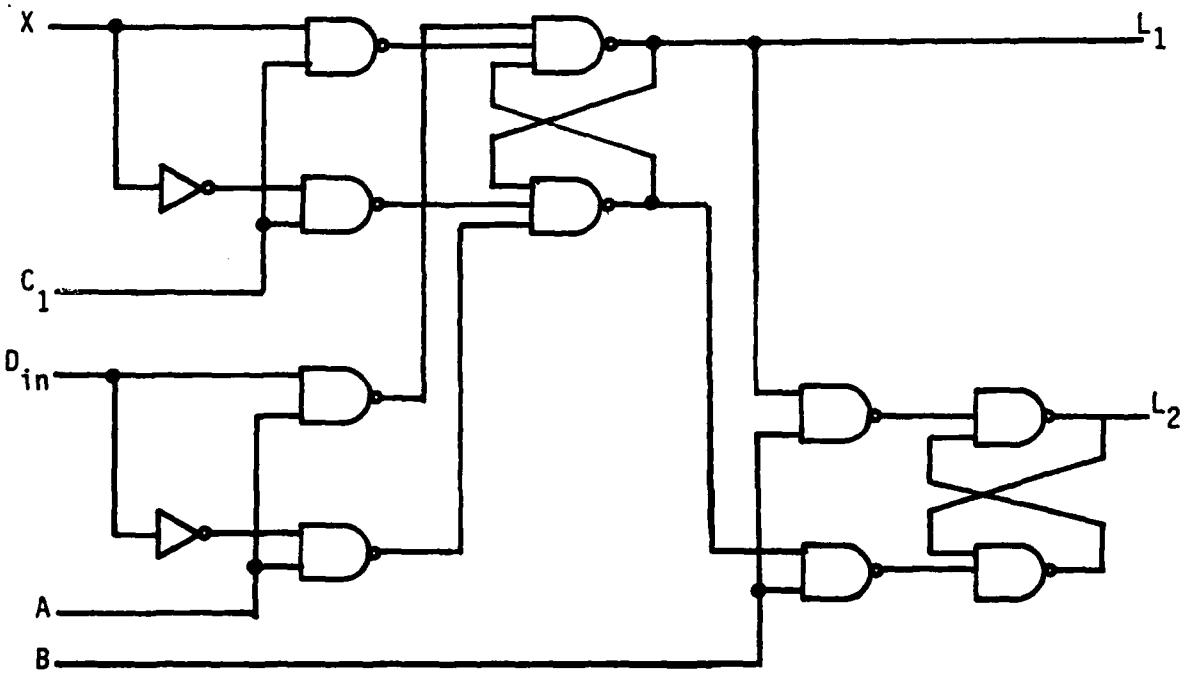
The main drawback of this design methodology is the fact that the scan-in and scan-out steps are relatively slow. Testing time increases with the length of SR, hence the method tends to become unacceptably slow in a circuit with a large memory, e.g., a CPU containing a large scratchpad RAM.

Scan-in/scan-out methods have been known in various forms under various names for at least a decade [Trischler 1980]. The approach was first completely described in 1973 [Williams and Angell 1973]. It has come into widespread use with the advent of VLSI in the late 1970s, particularly among mainframe computer manufacturers. Among the better-known implementations of this approach are Sperry Univac's SCAN/SET [Stewart 1978], Nippon Electric's SCAN-PATH [Funatsu et al. 1978], and IBM's LSSD (Level Sensitive Scan Design) [Eichelberger and Williams 1978, Berglund 1979]. The Selective Control Method discussed in Sec. 3.2 employs a similar testing philosophy. LSSD combines the basic scan-in/scan-out technique with the use of memory elements and a clocking mechanism that eliminate most timing problems, hence we consider LSSD here in somewhat more detail.

The memory part of an LSSD circuit consists of storage elements called shift-register latches (SRLs). An SRL is composed of two clocked D-type flip-flops, L_1 and L_2 interconnected as shown in Fig. 4.4. L_1 is the "system" latch and corresponds to a normal flip-flop in non-LSSD designs. L_2 is added to act as an intermediate storage element during the shifting operations associated with scan-in and scan-out. Alternatively, L_1 and L_2 may form a single master-slave flip-flop. A and B are non-overlapping clock



(a)



(b)

Fig. 4.4. (a) Symbol for SRL. (b) Logic design of SRL.

signals which control the operation of the SRL. The design of the SRL and the rules for the clock signals are such that the system is level sensitive, i.e., its behavior is independent of circuit and wire delays within the system. It is therefore free of hazards and other error-causing conditions. A circuit composed of combinational modules and SRLs is called an LSSD circuit if it obeys certain design rules [Eichelberger and Williams 1978] which ensure that

(1) The circuit as a whole is level-sensitive.

(2) The SRLs can be linked to form a shift register SR for scan-in/out operations.

Figure 4.5 shows a typical LSSD-based circuit. During normal operation the L_1 and L_2 latches forming each SRL act like a master-slave flip-flop controlled by the non-overlapping clocks C_1 and C_2. During scan-in/scan-out operations, each L_1 is linked serially to the corresponding L_2; while L_2 is linked to the L_1 latch in the preceding SRL. Thus the L_i's are connected to form a shift register SR with the serial data input/output lines D_{in} and D_{out}. Data is shifted into or out from SR by means of non-overlapping shift signals applied to the A and B control lines.

4.3 Bit-slice Design

Scan-in/scan out can be regarded as a method of imposing a regular structure on an otherwise irregular design. Another methodology that produces highly regular logic design is bit slicing [Hayes 1981]. The term bit-sliced is most often applied to microprocessors and similar circuits that have the following characteristics:

(1) The basic module or (bit) slice S_m performs a specified set of operations F on operands or data words of length m bits where $m \geq 1$.

(2) A set of n copies of S_m can be connected in the form of a 1-dimensional array or cascade as depicted in Fig. 4.6 so that the resulting bit-sliced system S_m^n performs the same set of operations F on mn-bit words.

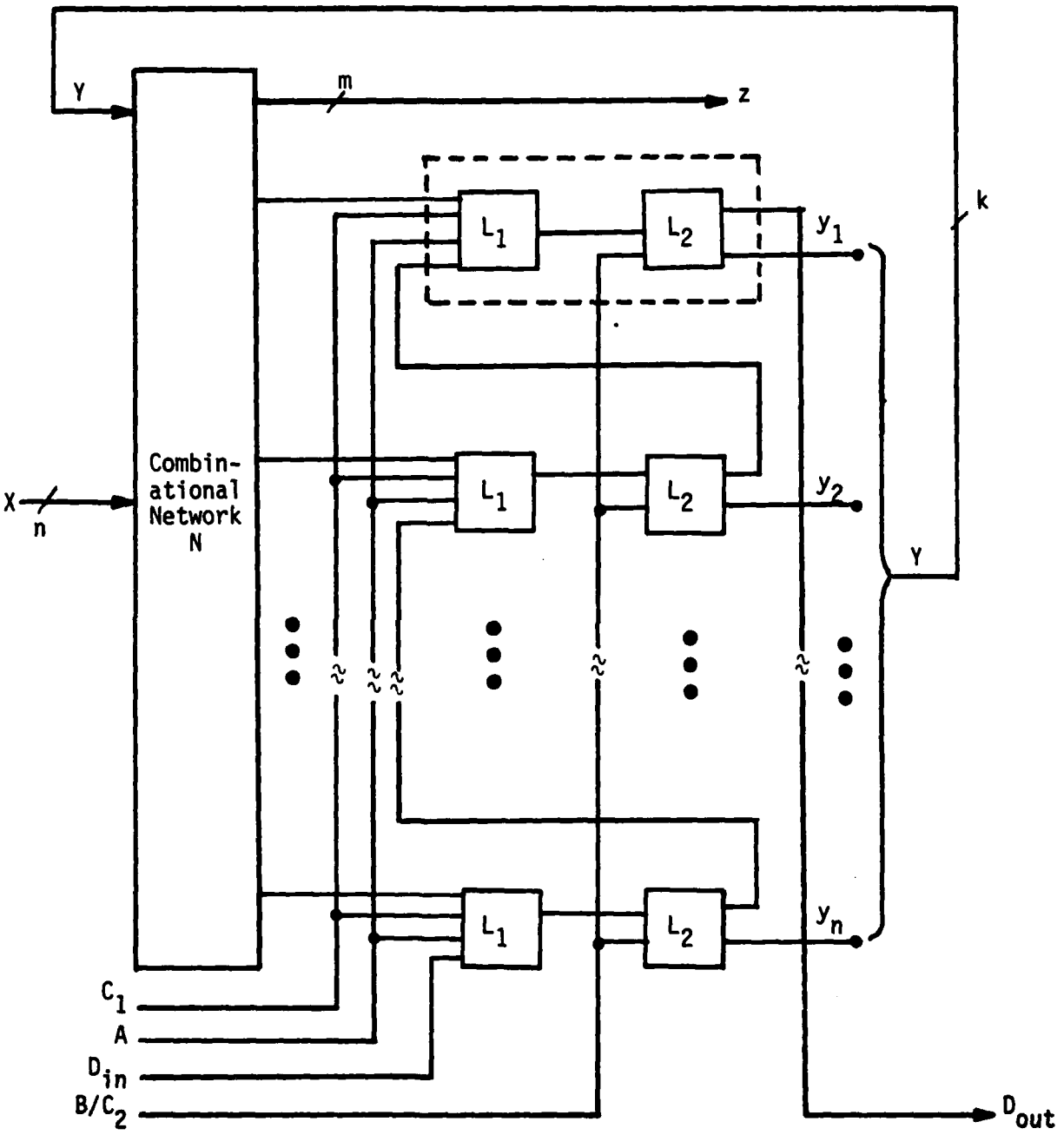


Fig. 4.5. A circuit designed using LSSD.

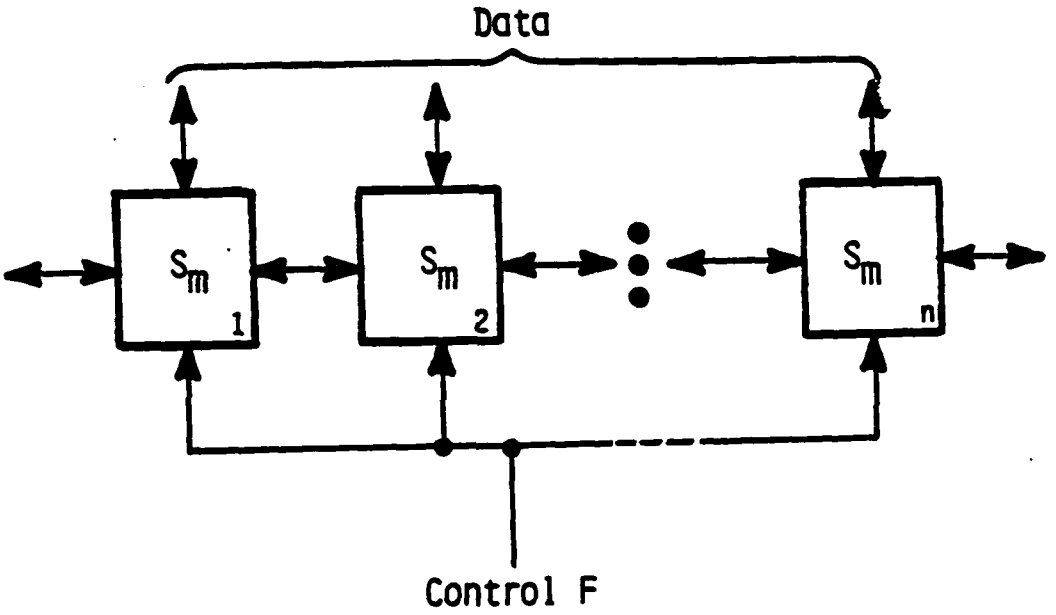


Fig. 4.6. General structure of a bit-sliced system S_m^n composed of n m-bit sliced S_m

Bit slicing has several advantages from a testability point of view. The individual slices are relatively simple; for example, a 4-bit word size is typical of microprocessor slices. This makes it feasible to generate test sets for individual slices that have very high fault coverage, even when relatively complex functional fault models are used [Sridhar and Hayes 1979, 1981]. Furthermore, the regularity of bit-sliced arrays can be exploited to derive tests for the complete array from those of a component slice. The first application of bit slicing to enhance testability was msde almost 20 years ago [Forbes et al. 1965]. However the use of bit-slicing has become widespread only in the last few years, since the appearance of the Advanced Micro Devices 2900 series of bit-sliced components [Mick and Brick 1980].

Figure 4.7 shows a model C of a general-purpose processor slice developed for the study of fault diagnosis in bit-sliced systems [Sridhar and Hayes 1981]. It differs from current commercial processor slices primarily in the fact that its word size is just one bit. (Note, however, that 1-bit non-bit-sliced microprocessors such as the Motorola 14500 are commercially available [Motorola 1977].) Test data for a one-bit slice of this kind can readily be extended to larger slices. The overall structure of C is very similar to that of the 2901 processor [Mick and Brick 1980].

For testing purposes C is viewed as a network of a small number of register-level modules $\{M_i\}$ such as multiplexers, registers and a combinational ALU. Consider the task of generating a test set T_C for all functional faults in C. Let T_i be a test set that detects all functional faults in an isolated module M_i. If M_i is combinational, then it is necessary and sufficient for T_i to be the set of all 2^n inputs to the module, where n is the number of distinct input lines of M_i. If M_i is a sequential circuit, then the checking sequence approach is used in constructing the test set T_i [Breuer and Friedman 1976]. This approach has been shown to yield test sequences of minimal or near-minimal length in the case of the small sequential modules considered here. When M_i is a component of C, faults in M_i are detected by a set T_i^*, which when applied to the primary inputs of C, causes T_i to be applied to M_i, and causes the responses of M_i to be propagated to the observable outputs of C. Since at most one module is allowed to be faulty, a composite test set for the entire circuit is obtained by

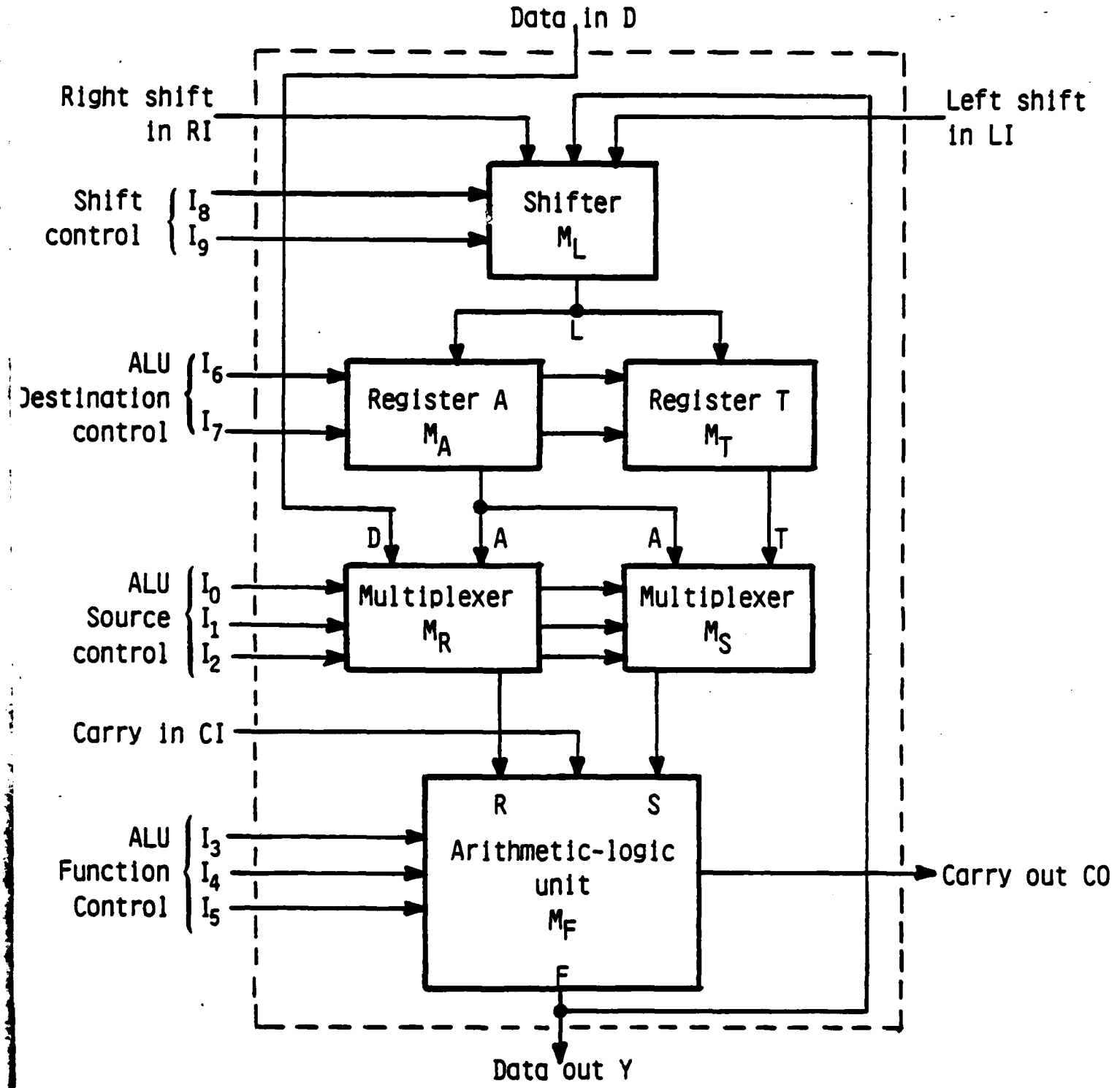


Fig. 4.7. An easily-testable 1-bit processor slice C

combining all the T_i^*, many of which contain common test patterns. It is not difficult to show that C can be tested by a test set T_C containing 114 test patterns. Note that this is less than twice the minimum number of tests (64) required to test the ALU module M_F alone.

N copies of C can be cascaded as in Fig. 4.6 to form an N-bit processor array $C_0, C_1, \ldots, C_{N-1}$. In order to test an array of this type, it is necessary to apply the test set T_C to each slice C_j in the array irrespective of its position. In addition, the responses of C_j to T_C must be propagated to the observable outputs of the array, which comprise the lines Y_0:Y_{N-1} and the array carry out line. Most of the tests of the form T_i^* for a module M_i in C can be applied simultaneously to every copy of C in the array since they do not affect the interslice shift and carry lines. Certain test patterns involving the shift and carry lines cannot be applied simultaneously to every slice; for instance, a test that results in CI $\neq$ CO. Such cases can be covered by applying on type of test pattern to even-numbered slices, and another to odd-numbered slices. These test patterns can be so ordered that the entire array is completely tested using the same number of tests as used for C alone. Moreover, the tests for the array are easily computed from the tests T_C for C. Consequently, testing a bit-sliced array of arbitrary length constructed from C is only a little more difficult than testing C itself, and the testing time is constant. The 1-bit slice C of Fig. 4.7 can be extended to more closely resemble commercial bit sliced without destroying its desireable testing properties.

Bit-sliced systems of the foregoing type have the very desirable properties that comprehensive test sets are easy to compute, and that test generation complexity is, at least to a first approximation, independent of word size. Thus the fundamental property of bit-sliced systems is that their functional behavior is independent of word size can also be extended into the realm of test generation. It should be noted, of course, that commercial processor slices are often more complex than C and their test generation problems are correspondingly more difficult. The significance of the C model is that it demonstrates that bit slicing can be used to build realistic processors that are far easier to test than equivalent non-bit-sliced processors.

4.4 Summary and Evaluation

Design for testability involves various tradeoffs between testability, performance and hardware costs. A general design method must provide a high degree of fault detectability and resolution with a relatively low overhead in added logic and IO pins. One of the few design approaches that meets these goals is scan-in/scan-out, of which IBM's LSSD technique is the best-known example. The hardware overhead associated with LSSD is small, typically from five to ten percent of the equivalent non-LSSD design. Its use of serial access to the memory elements implies the need for very few extra IO pins, an especially important consideration in VLSI designs. The design rules for LSSD are relatively simple and have minimal impact on system performance during normal operation. The major advantage of LSSD from testability viewpoint, is that the test pattern generation problem is reduced to the relatively simple one of testing a combinational circuit. LSSD circuits are also very suitable for delay fault testing [Lesser and Shedletsky 1980]. The serial nature of the scan-in/scan-out process makes testing relatively slow, and limits the number of memory elements that may be used. LSSD is therefore not applicable to designs containing more than a few hundred memory elements. Scan-in/scan-out may also be unsuitable for VLSI designs because of the difficulty of synchronizing extermely long shift-registers [Mead and Conway 1980].

Bit slicing is a more specialized design technique that also leads to easily testable circuits. The simplicity of the individual slices allows complex fault models to be used, while the regular structure of bit-sliced arrays makes it easy to extend test sets from a single slice to an entire array. In some cases this extension can be made with little or no increase in the number of test patterns required. Bit-slicing is well suited to the design of processors and memories; unlike LSSD it is unsuitable for designing unstructured "random" logic.

5. BIBLIOGRAPHY

[Anon 1980] Anon. "A study of testability standardization for electronic systems and equipment," U.S. Navy Internal Report, Fall 1980.

[Azéma et al. 1978] P. Azéma, A. Lozes & M. Diaz. "Test point placement for combinational circuits," *Digital Processes*, vol. 3, pp. 227-235, 1977.

[Berglund 1979] N.C. Berglund. "Level-sensitive Scan Design tests chips, boards, system," *Electronics*, vol. 52, no. 6, pp. 108-110, March 15, 1979.

[Bennetts & Scott 1976] R.G. Bennetts & R.V. Scott. "Recent developments in the theory and practice of testable logic design," *Computer*, vol. 9, no. 6, pp. 47-63, June 1976 (reprinted from *Radio and Electronic Engineer*, vol. 45, pp. 667-679, Nov. 1975, updated and revised).

[Bozorgui-Nesbat & McCluskey 1980] S. Bozorgui-Nesbat & E.J. McCluskey. "Structured design for testability to eliminate test pattern generation," *Digest 10th Fault-Tolerant Computing Symposium*, Kyoto, pp. 158-163, October 1980.

[Breuer & Friedman 1976] M.A. Breuer & A.D. Friedman. *Diagnosis and Reliable Design of Digital Systems*, Woodland Hills, California, Computer Science Press, 1976.

[Chang & Heimbigner 1974] H.Y. Chang & G.W. Heimbigner. "LAMP: Controllability, observability and maintenance engineering technique (COMET)," *Bell Sys. Tech. Journ.*, vol. 53, pp. 1505-1534, October 1974.

[Chiang & McCaskill 1976] A.C.L. Chiang & R. McCaskill. "Two new approaches simplify testing of microprocessors *Electronics*, vol. 49, no. 2, pp. 100-105, January 22, 1976.

[Consolla & Danner 1980] W.M. Consolla & F.G. Danner. "An objective printed circuit board testability design guide and rating system," Rome Air Dev. Center, Tech. Rept. RADC-TR-79-327, January 1980.

[Dussault 1978] J.A. Dussault. "A testability measure," *Proc. 1978 Semiconductor Test Conference*, pp. 113-116, November 1978.

[Eichelberger & Williams 1978] E.B. Eichelberger & T.W. Williams. "A logic design structure for LSI testability," *Journ. Design Autom. and Fault-Tolerant Computing*, vol. 2, no. 2, pp. 165-178, May 1978.

[El-Ziq & Su 1977] Y.M. El-Ziq & S.Y.H. Su. "Design and testing of diagnosable MOS logic networks, *COMPCON Digest*, pp. 254-260, Spring 1977.

[Fairchild 1980] Fairchild. "F3870 single-chip microcomputer," Data Sheet, Fairchild, Mountain View, California, May 1980.

[Friedman 1973] A.D. Friedman. "Easily testable iterative systems," *IEEE Trans. on Computers*, vol. C-22, pp. 1061-1064, December 1973.

[Fox 1977] J.R. Fox. "Test point condensation in the diagnosis of digital circuits, *Proc. IEE (London)*, vol. 124, no. 2, pp. 89-94, February 1977.

[Funatsu et al. 1978] S. Funatsu, N. Wakatsuki & A. Yamada. "Designing digital circuits with easily testable consideration," *Digest 1978 Semiconductor Test Conf.*, pp. 98-102, October 1978.

[Goldstein 1979] L.H. Goldstein. "Controllability/observability analysis of digital circuits," *IEEE Trans. Circuits & Systems*, vol. CAS-26, pp. 685-693, September 1979.

[Goldstein & Thigpen 1980] L.H. Goldstein & E.L. Thigpen. "SCOAP: Sandia controllability observability analysis program," *Proc. 17th Design Automation Conference*, Minneapolis, pp. 190-196, June 1980.

[Grason 1979] J. Grason. "TMEAS: a testability measurement program," *Proc. 16th Design Automation Conference*, San Diego, pp. 165-161, June 1979.

[Grason & Nagle 1980] J. Grason & A.W. Nagle. "Digital test generation and design for testability," *Proc. 17th Design Automation Conference*, Minneapolis, pp. 175-189, June 1980.

[Hayes 1981] J.P. Hayes. "A survey of bit-sliced computer design," *Journ. of Digital Systems*, 1981, to appear.

[Hayes & McCluskey 1980] J.P. Hayes & E.J. McCluskey. "Testability considerations in microprocessor-based design," *Computer*, vol. 13, no. 3, pp. 17-26, March 1980.

[Hewlett-Packard 1977] Hewlett-Packard. "Designing digital circuits for testability," Applic. Note 210-4, Palo Alto, California, January 1977.

[Hsu et al. 1978] F. Hsu, P. Solecky & L. Zobniw. "Selective controllability: a proposal for testing and diagnosis," *Proc. 15th Design Automation Conference*, Las Vegas, pp. 110-116, June 1978.

[Lesser & Shedletsky 1980] J.D. Lesser & J.J. Shedletsky. "An experimental delay test generator for LSI logic," *IEEE Trans. on Computers*, vol. C-29, pp. 235-248, March 1980.

[Mancone 1979] J. Mancone. "Testability guidelines," *Electronics Test*, vol. 2, pp. 14-16, March 1979.

[McCluskey 1978] E.J. McCluskey. "Design for maintainability and testability," *Proc. Government Microcircuit Applications Conference (GOMAC)*, Monterey, California, pp. 44-47, November 1978.

[Mead & Conway 1980] C. Mead & L. Conway. *Introduction to VLSI Systems*. Reading, Massachusetts, Addison-Wesley, 1980.

[Mick & Brick 1980] J. Mick & J. Brick. *Bit-slice Microprocessor Design*, New York, McGraw-Hill, 1980.

[Motorola 1977] Motorola. *MC14500B Industrial Control Unit Handbook* (by V. Gregory & B. Dellande), Motorola, Phoenix, 1977.

[Muehldorf 1976] E.I. Muehldorf. "Designing LSI logic for testability," *Digest 1976 Semiconductor Test Symp.*, Cherry Hill, New Jersey, pp. 45-49, October 1976.

[Parker & McCluskey 1975] K.P. Parker & E.J. McCluskey. "Probabilistic treatment of general combinational networks," *IEEE Trans. Computers*, vol. C-24, pp. 668-670, June 1975.

[Reddy 1972] S.M. Reddy. "Easily testable realizations for logic functions," *IEEE Trans. on Computers*, vol. C-21, pp. 1183-1188, November 1972.

[Roth 1980] J.P. Roth. *Computer logic, testing and verification*, Potomac, Maryland, Computer Science Press, 1980.

[Rutman 1972] R.A. Rutman. "Fault detection test generation for sequential logic by heuristic tree search," IEEE Computer Group Repository, Paper No. R72-187, 1972.

[Saluja 1978] K.K. Saluja. "A testable design of sequential machines," *Digest 8th Conference on Fault-Tolerant Computing*, Toulouse, pp. 185-190, June 1978.

[Savir 1980] J. Savir. "Syndrome testable design of combinational circuits," *IEEE Trans. Computers*, vol. C-29, pp. 442-451, June 1980.

[Sridhar & Hayes 1979] T. Sridhar & J.P. Hayes. "Testing bit-sliced microprocessors," *Digest 9th International Symposium on Fault-Tolerant Computing*, Madison, Wisconsin, pp. 211-218, June 1979.

[Sridhar & Hayes 1981] T. Sridhar & J.P. Hayes. "A functional approach to testing bit-sliced microprocessors," *IEEE Trans. on Computers*, 1981, to appear.

[Stephenson & Grason 1976] J.E. Stephenson & J. Grason. "A testability measure for register transfer level digital circuits," *Proc. 6th Symposium on Fault-Tolerant Computing*, Pittsburgh, pp. 101-107, June 1976.

[Stewart 1978] J.H. Stewart. "Application of scan/set for error detection and diagnostics," *Proc. 1978 Semiconductor Test Conference*, pp. 152-158, October 1978.

[Trischler 1980] E. Trischler. "Incomplete scan path with an automatic test generation methodology," *Digest 1980 Test Conference*, Philadelphia, pp. 153-162, November 1980.

[Susskind 1981] A.K. Susskind. "Testability and reliability of LSI," Rome Air Dev. Center, Final Tech. Rept. RADC-TR-80-384, January 1981.

[Williams & Angell 1973] M.J.Y. Williams & J.B. Angell. "Enhancing testability of large-scale integrated circuits via test points and additional logic," *IEEE Trans. on Computers*, vol. C-22, pp. 46-60, January 1973.

[Williams & Eichelberger 1977] T.W. Williams & E.B. Eichelberger. "Random test patterns within a structured sequential logic network," *Digest Semiconductor Test Symposium*, Cherry Hill, New Jersey, pp. 19-27, October 1977.

[Williams & Parker 1979] T.W. Williams & K.P. Parker. "Testing logic networks and designing of testability," *Computer*, vol. 12, no. 10, pp. 9-21, October 1979.

END

DATE
FILMED

7-83

DTIC